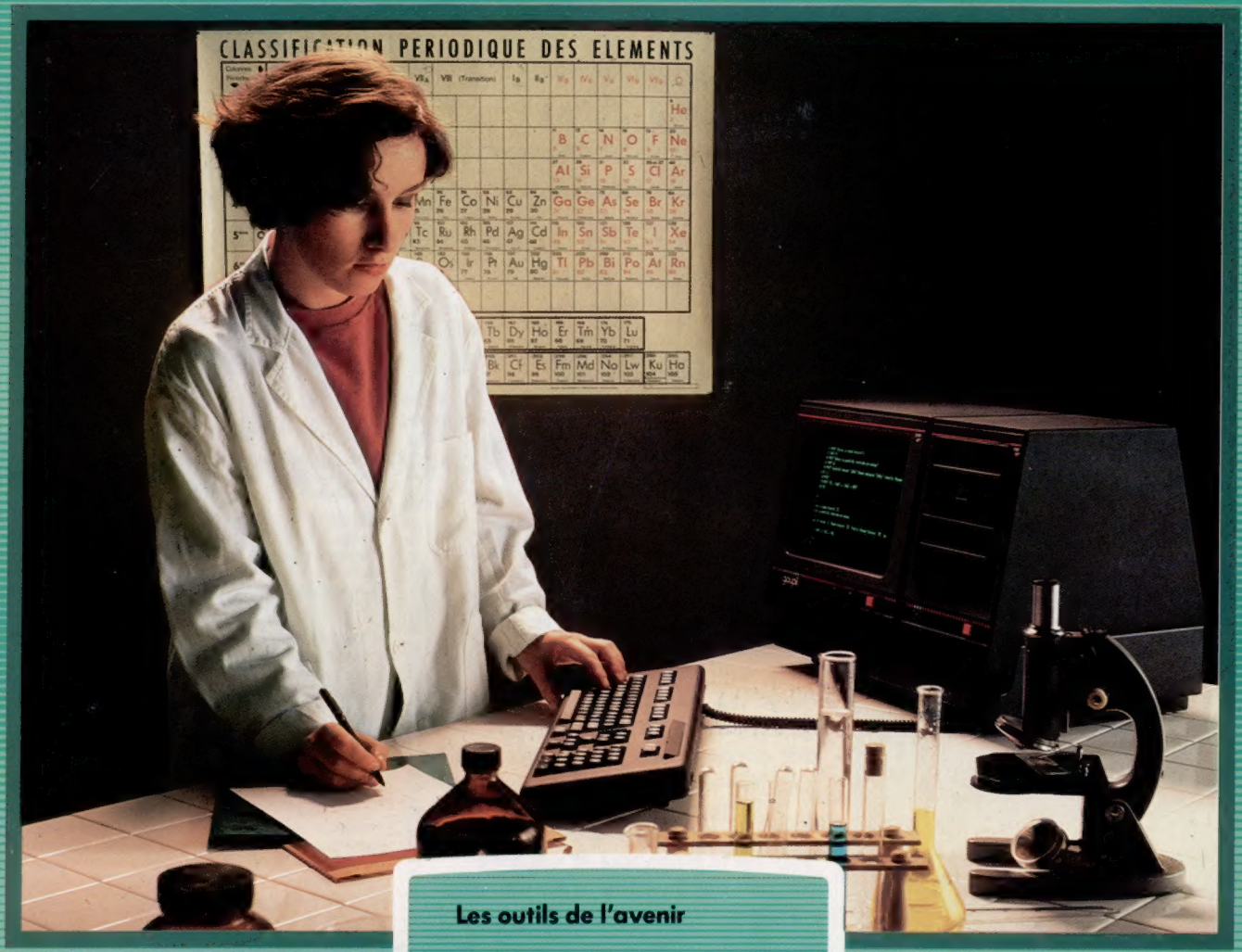
ABC INFORMATIQUE

N°25

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE



Les outils de l'avenir

Algèbre de Boole

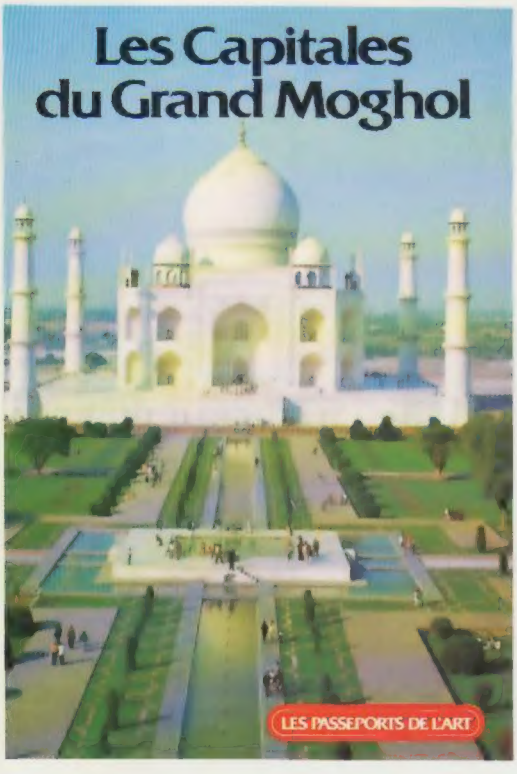
Le basic de Spectrum

Jeux : batailles spatiales

EDITIONS ATLAS

M6062-25-12F 85FB-3,80FS-$1.95

Dans toutes les librairies



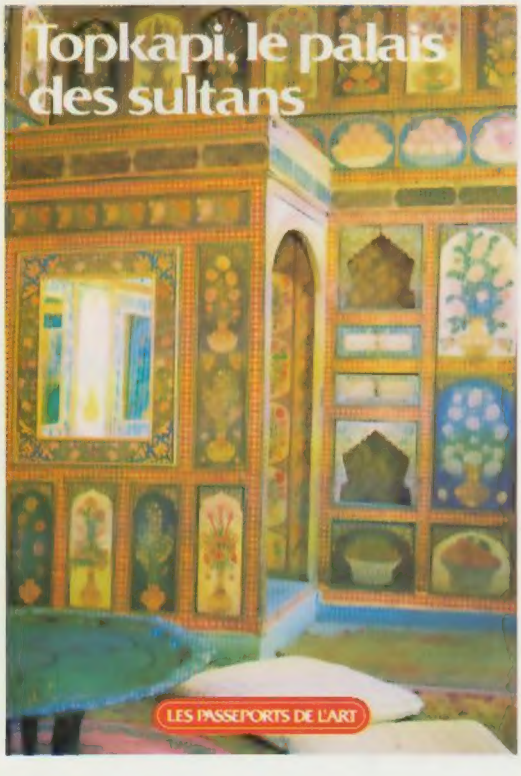
Les passeports de l'art

Les capitales du Grand Moghol

Avec l'arrivée au pouvoir des souverains moghols, l'Inde, du XVI^e^ au XVIII^e^ siècle, s'ouvrit à la civilisation islamique. Agra, capitale de l'empereur Akbar le Grand, et Fatehpur Sikri, aujourd'hui cité fantôme, devinrent les principaux centres politiques de l'Empire. Les magnifiques monuments indo-musulmans de l'Inde, dont le Tadj Mahall, merveilleux mausolée de marbre blanc, est le plus beau fleuron, témoignent d'une gloire passée.

Un volume relié.
Couverture cartonnée. 76 pages.
71 photos en couleurs.
Format : 17 × 24 cm.

Dans toutes les librairies



Les passeports de l'art

Topkapi, le palais des sultans

Fondé en 1460, le palais de Top Kapu Saray (la porte du Canon), connu aussi sous le nom de Topkapi, devint la résidence favorite des sultans d'Istanbul au XVII^e^ siècle. Ensemble de bâtiments, kiosques et pavillons alternant avec des cours et des jardins élégants, s'étendant de la Corne d'Or à la basilique Sainte-Sophie, Topkapi, cité dans la cité, domine le Bosphore et exprime par son architecture le croisement de deux mondes.

Un volume relié.
Couverture cartonnée. 76 pages.
70 photos en couleurs.
Format : 17 × 24 cm.

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume III, n° 25.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : S.M.T.-Goupil.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : juin 1984. 22846. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Paré pour l'avenir

La plupart des adultes éprouvent de grandes difficultés à apprendre une matière nouvelle; ABC Informatique est conçu pour simplifier leur apprentissage.

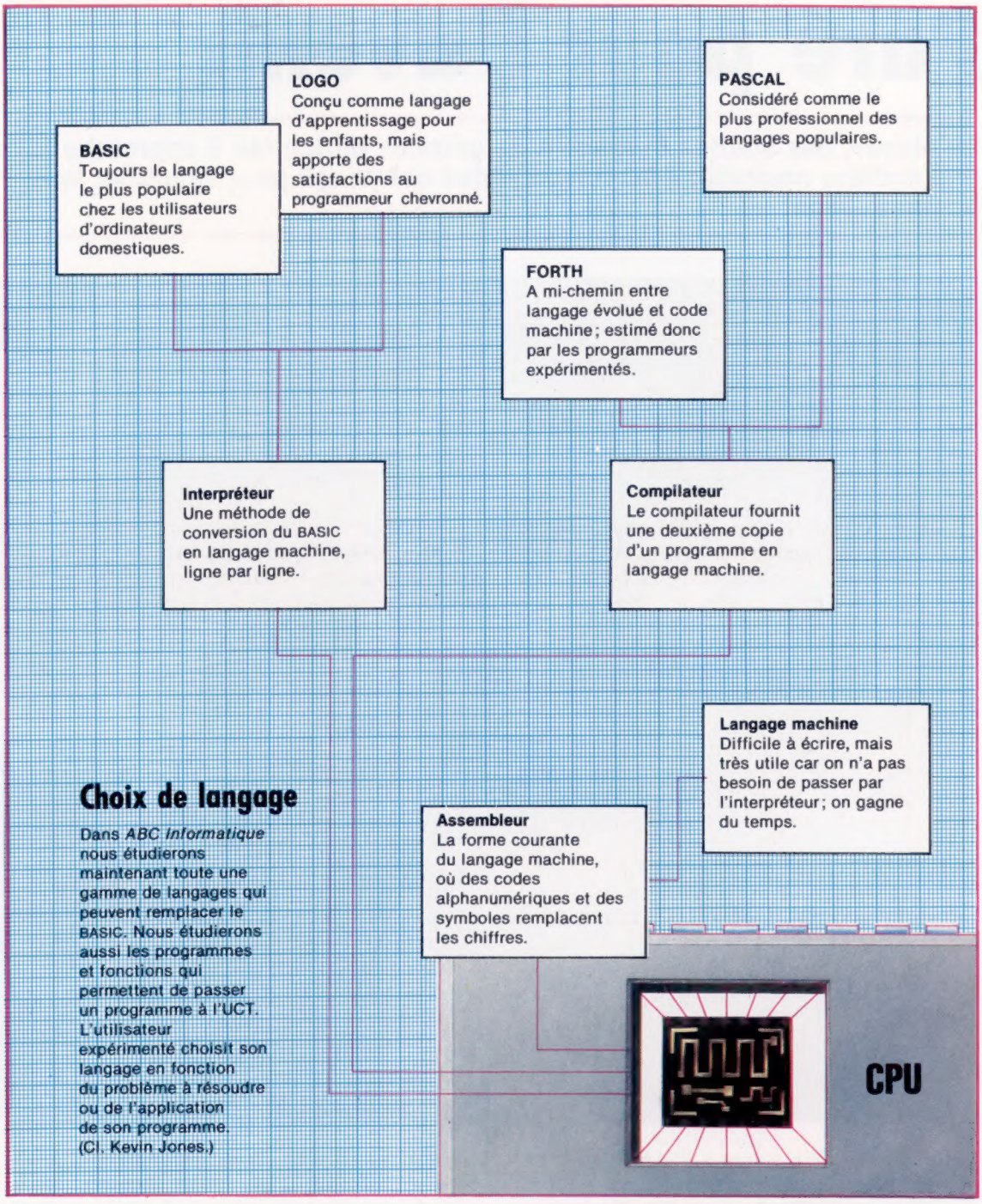
Les deux premiers volumes d'*ABC Informatique* ont permis de se familiariser avec le monde des micro-ordinateurs, avec des programmes simples, avec quelques données générales d'approche du marché. Il est utile, en ouvrant ce troisième volume de l'encyclopédie, de refaire le point pour vous aider à dresser le bilan de vos besoins particuliers, et pour vous présenter, dans les grandes lignes, le contenu des volumes à venir.

Par où commencer? Par l'achat d'un ordinateur bon marché? Il y en a tant; on aura du mal à choisir sans quelques conseils éclairés. Et ensuite? La machine offre peut-être plus d'un langage; lequel correspond le mieux aux besoins de l'utilisateur? Parmi les logiciels, lesquels seront les plus avantageux? Faut-il les prendre en cartouche ou sur bande? Doit-on acquérir un lecteur de disquettes, ou une cassette suffira-t-elle? A-t-on besoin d'une imprimante tout de suite, ou ne vaut-il pas mieux attendre que le prix des imprimantes plus évoluées baisse? Si vous avez des enfants, comment concilier leur désir d'utiliser sans arrêt des jeux avec votre sentiment qu'ils devraient se servir de l'ordinateur de façon plus sérieuse?

L'informatique a progressé à une telle vitesse, avec des conséquences si évidentes pour notre vie de tous les jours, que nous sommes tentés de l'entourer d'un certain mystère. Il n'est certes pas facile d'accepter une force dont l'action est invisible et apparemment incompréhensible.

Il est vrai que de plus en plus de personnes sont formées pour utiliser des terminaux d'ordinateurs. Mais il y a une énorme différence entre la formation et l'éducation, entre la maîtrise d'une tâche répétitive et la compréhension globale d'un système, ses possibilités et ses limites.

Tenez-vous au courant
En moins d'une dizaine d'années, l'ordinateur est devenu partie intégrante de notre société. Chaque année les transformations se font plus importantes. Il faut un certain effort pour se tenir au courant — et pour le nouveau venu la tâche est encore plus difficile. Un cours bien conçu, progressif, à suivre chez soi, est une aide précieuse. (Cl. Paul Chave.)



Les notices livrées avec les machines n'apportent qu'une solution partielle : elles ne donnent jamais une vision globale, et ne permettent pas de comparer les avantages et les inconvénients de tous les matériels disponibles. Après tout, quel fabricant donnerait une publicité gratuite aux réalisations de ses concurrents?

Il est déjà difficile pour les professionnels de l'informatique — programmeurs et analystes en particulier — de se maintenir au courant des dernières découvertes en la matière, même si ces personnes peuvent compter sur le milieu dans lequel ils travaillent. Les fournisseurs d'équipements informatiques et les recyclages payés par l'entreprise qui les emploie viennent à point, normalement, pour remettre systématiquement en cause leur niveau de connaissance. Le problème est tout à fait différent lorsqu'il s'agit pour les amateurs ou les passionnés d'informatique d'obtenir la bonne information. *A priori,* ils ne peuvent compter que sur eux-mêmes, leur temps et leur argent, pour aller à la recherche du « tuyau » impartial. Ce n'est pas évident pour tout le monde, alors, comment faire?

La solution, semble-t-il, réside dans un cours d'informatique, bien structuré, progressif, mais à la portée de tous dès le départ. Un tel cours, sous forme de méthode à suivre chez soi, et complété peut-être par des stages ou des cours du soir, représente une façon pratique et peu onéreuse d'obtenir une bonne formation en informatique. Tel est, du moins, l'avis de ceux

qui s'occupent déjà d'informatique dans un contexte industriel ou universitaire.

Ce cours a pour but, non seulement d'initier les étudiants à la programmation et à l'utilisation d'un ordinateur domestique, mais aussi de leur donner un aperçu des multiples applications des ordinateurs dans la vie de tous les jours. Il passe en revue toutes les machines qui existent, présente les périphériques et accessoires disponibles, et en explique les principes de fonctionnement. Pour situer l'informatique dans son contexte, il faut étudier en profondeur ses applications, et les logiciels qui rendent ces applications possibles. Enfin, le cours aborde quelques notions de logique formelle, de mathématiques, et de l'histoire de l'informatique. Bref, notre cours couvre tous les sujets qui seraient traités par un programme d'université ou d'I.U.T.

Notre intention, dans *ABC Informatique,* est de vous familiariser le plus rapidement possible avec tous ces éléments. Partant d'un niveau moyen en BASIC et d'une petite expérience en graphisme et en synthèse de son, vous allez vous initier aux autres langages évolués utilisés dans les mini-ordinateurs — PASCAL, FORTH, LOGO par exemple — et prendre contact avec la programmation en langage machine, clé qui ouvre les portes de l'UCT.

Connaissant le langage machine, on peut étudier la conception des langages évolués. Ensuite, ayant vu comment fonctionnent les compilateurs et les interpréteurs, on peut mettre ensemble toutes ces connaissances pour définir son propre langage et commencer à en écrire le compilateur.

Toutefois, le BASIC ne sera pas négligé. Nous allons étudier les subtilités de ce langage, en travaillant sur des projets de création de logiciels et de jeux d'aventures.

Un système valable de traitement

Au-delà du fonctionnement interne de l'ordinateur, nous allons examiner les méthodes de gestion de fichiers, aussi bien sur bande que sur disquette, en nous servant de l'étude déjà faite, sur les structures de la mémoire interne de l'ordinateur. Ainsi, nous pourrons élargir les capacités du plus petit ordinateur domestique pour en faire un système valable de traitement de données.

Bien sûr, il ne suffit pas d'étudier la machine toute seule. Nous ferons un examen approfondi de toute la gamme de progiciels disponibles — systèmes de traitement de texte, de gestion de fichiers, etc. — afin de comprendre leur fonctionnement et aussi pour voir comment travaillent les programmeurs professionnels; leurs techniques peuvent nous être utiles pour nos propres programmes.

L'électronique de base sera traitée également : fonction et conception des composants, façon dont ils se combinent dans les ordinateurs et les périphériques. Et, naturellement, nous nous pencherons sur les machines elles-mêmes. Bien sûr, nous nous contenterons de détailler les mini-ordinateurs dont les prix sont compatibles avec une bourse familiale, même si nous devons aborder parfois des éléments d'ordinateurs plus importants. Nous décrirons également leurs périphériques, leur prix et le travail dont ils sont capables, leur impact sur le monde de l'informatique.

Cela sans négliger le côté humain de l'industrie : il y aura une place dans le cours pour les hommes qui conçoivent les logiciels et construisent les machines, et même pour les utilisateurs qui ont fait avancer la science.

Si vous vous intéressez à l'informatique avec l'idée d'en faire votre métier, alors un cours à suivre chez soi peut utilement remplacer les premières étapes d'une formation professionnelle. En permettant à chacun d'avancer à son propre rythme, il sert aussi bien l'étudiant doué que celui qui a besoin d'un peu plus de temps pour se familiariser avec ce sujet complexe.

Enfin, si vous désirez simplement être mieux informé sur une technologie qui est appelée à transformer le monde dans lequel vous vivez, *ABC Informatique* guidera vos recherches. En plus des principes de base de l'informatique, nous étudierons l'impact de la nouvelle technologie sur la société tout entière. Comment l'informatique changera-t-elle les relations sociales? Quelles seront les conséquences politiques de l'« explosion de l'information » que le microprocesseur bon marché est en train de déclencher? Il est difficile de trouver des réponses à ces questions. Les médias les traitent de façon superficielle, tandis que beaucoup d'ouvrages sur l'informatique tendent vers une complexité inutile. *ABC Informatique* vous exposera les données du problème afin que vous puissiez fournir votre propre réponse.

Le bond en avant
Présenté à la presse mondiale au début de 1984, et prévu sur le marché au cours du printemps, le Quantum Leap de Sinclair rompt enfin avec le microprocesseur Z80, qui est remplacé ici par une adaptation du 68000 32 bits de Motorola. Il possède 128 K de RAM (extension de 512 K possible) et deux Microdrives QL intégrés. Sinclair abandonne aussi son BASIC à une seule touche. (Cl. Ian McKinnell.)

Meilleur choix

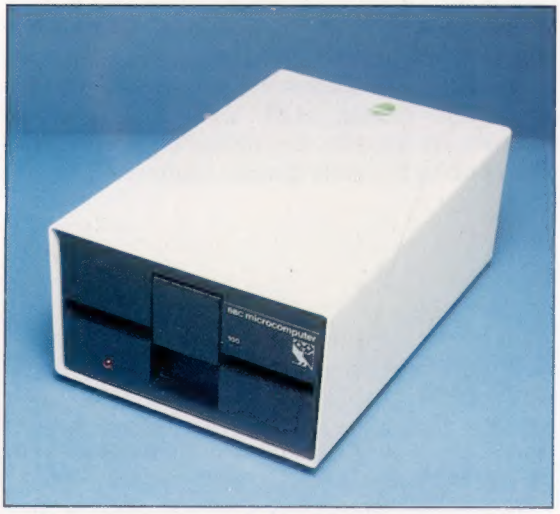
Jusqu'à une époque récente, les lecteurs pour les différentes formes de disques souples étaient trop chers pour l'amateur moyen. Mais grâce aux progrès technologiques les prix ont baissé.

Les ordinateurs traitent des informations avec une souplesse extraordinaire. Mais il ne sert à rien de traiter l'information si l'on ne peut pas la stocker en cours de travail ou quand on arrête sa machine. Il existe plusieurs façons d'y parvenir. Avant d'entrer dans les détails, rappelons pour ceux qui ne le sauraient déjà — ils sont peu nombreux, assurément — qu'il existe deux types de mémoire : les ROM ou mémoires mortes dans lesquelles tout est inscrit une fois pour toutes sans possibilité d'effacement, et les RAM ou mémoires vives, dans lesquelles on peut effacer ce qui y est écrit. Les amateurs expérimentés connaissent déjà les limites des cartouches ROM et des cassettes ordinaires. Ils ont envie d'essayer les possibilités plus grandes des disques magnétiques. N'oublions pas que ce sont les créateurs de l'Apple qui ont, les premiers, mis au point des lecteurs de disquettes performants et à prix abordables.

ATTENTION DANGER

Il ne faut jamais mettre un disque souple à côté d'un aimant. Même votre téléphone comporte des électro-aimants (qui servent à faire marcher la sonnerie), et les baffles de votre chaîne hi-fi contiennent des aimants très puissants.

Mais avant d'étudier les disques, jetons un coup d'œil d'abord sur les autres systèmes qui font toujours le bonheur des utilisateurs.



Unité de disquettes BBC
Avant d'utiliser un lecteur de ce type avec le BBC modèle B, il faut que la ROM du DOS (système d'exploitation du disque) soit installée dans la machine elle-même. Les lecteurs « intelligents », par contre, sont équipés d'une puce DOS intégrée. (Cl. Ian McKinnell.)

Cartouches

Ce moyen de stockage n'est pas très utile au programmeur. La plupart des cartouches ne contiennent qu'un genre de PROM (mémoire morte programmable) qui sert surtout à entrer des données dans l'ordinateur, ou à apporter des possibilités supplémentaires, comme des extensions au BASIC intégré.

Certaines cartouches, cependant, renferment des PROM effaçables (EEPROM), où l'on peut écrire et lire comme sur la RAM interne, et qui sont « non volatiles » (c'est-à-dire que les données restent enregistrées quand on met la machine à l'arrêt). De même, quelques ordinateurs sont équipés de cartouches renfermant des puces mémoire vive en technologie CMOS (RAM CMOS) à basse consommation, qui peuvent conserver des données grâce à une petite pile logée dans la cartouche.

Mais le stockage à l'aide des EEPROM ou RAM CMOS revient cher : prenez en compte le fait que constituer une petite bibliothèque de ces cartouches coûte autant que l'achat d'un lecteur de disquettes approprié.

Cassettes

Adoptées dès le début, à cause du prix élevé des lecteurs de disquettes, les cassettes sont toujours appréciées pour leur faible coût et leur facilité d'emploi : la plupart des gens savent déjà s'en servir. Normalement, tout lecteur de cassette de qualité moyenne suffit; mais certains fabricants — Commodore et Atari, par exemple — insistent sur l'emploi d'une unité spéciale.

Les programmes et les données sont stockés en code binaire sous forme de fichiers séquentiels, enregistrés avec des tonalités particulières pour représenter les 0 et les 1. Habituellement, les données d'identification sont d'abord enregistrées (nom du fichier, peut-être l'adresse de la mémoire interne d'où il est sorti), suivies du fichier lui-même : un bit à la fois, en blocs de 1 octet qui sont formatés en segments de 256 octets. Dans beaucoup d'ordinateurs, un chiffre clé incorporé à chaque segment permet de vérifier qu'il n'y a aucune erreur de transcription.

Les commandes habituelles sont SAVE pour stocker les fichiers et LOAD pour les relire. Certains systèmes comportent d'autres possibilités : lecture de la bande pour sortir un catalogue de noms de fichiers, commandes spéciales pour stocker et rechercher différents types de données, etc.

A côté de son prix modique et de sa facilité d'emploi, la cassette présente toutefois quelques graves inconvénients :

1. La plupart du temps, l'utilisateur doit manipuler lui-même les commandes du lecteur : le positionnement de la bande et le réglage du volume sont critiques.

2. Les données étant stockées de façon séquentielle (sauf dans le cas du lecteur Hobbit, géré par le logiciel, et le lecteur intégré de l'Epson HX-20), la recherche d'un fichier particulier implique une surveillance attentive du

compteur (si compteur il y a!) par l'utilisateur, ou la lecture par la machine de toute la bande pour trouver le nom du fichier. De même, il est impossible, en stockage séquentiel, de travailler efficacement sur des données éparpillées à différents endroits tout au long d'un fichier. Ce genre de travail exige un système de stockage à accès aléatoire, qui devient essentiel quand il s'agit de gérer une base de données tels un listage d'adresses ou un système de gestion de stocks.

3. Ajoutons que la vitesse de lecture très réduite de la cassette (entre 300 et 1 200 bits/seconde) rend l'emploi d'un système de stockage sur bande extrêmement fastidieux. Charger un petit programme de quelque 5 K peut exiger de une à trois minutes. Il n'est donc pas commode de faire des copies supplémentaires des programmes, ce qui est pourtant à conseiller.

4. Les données, correctement enregistrées au départ, peuvent se dégrader à la longue à cause de l'usure des têtes de lecture/enregistrement.

5. Les magnétocassettes n'ont pas tous les mêmes caractéristiques; des données enregistrées sur une machine risquent de ne pas être relues par une autre. Et la mécanique rudimentaire de certaines machines peut endommager les bandes.

Disque souple

A côté du stockage sur bande ou en cartouche, le système à disque possède peu d'inconvénients majeurs. Les lecteurs de disques sont complexes et donc chers — à partir de 2 000 F. Et les disques eux-mêmes ne sont pas bon marché. Mais l'utilisateur retrouve un moyen de stockage fiable, souple et rapide, qui marche cent fois plus vite que la bande.

Tous les lecteurs de disquettes possèdent un système d'exploitation (DOS) qui renferme un programme pour formater les données sur les pistes de la disquette. Il y a normalement entre 35 et 80 pistes par face, chacune divisée en un certain nombre de segments; les pistes courtes vers le centre du disque comportent moins de segments que les pistes extérieures. Chaque segment contient un bloc de données, normalement de 256 octets.

Le DOS enregistre la location de chaque élément d'information, grâce à un catalogue et un répertoire de disponibilité des blocs (BAM) stockés sur le disque même ou dans la mémoire centrale. Le BAM tient un registre des blocs en occupation et de ceux qui restent libres. Le catalogue enregistre le nom, le type et la location par piste et par segment de chaque fichier. Il est situé d'habitude sur la piste centrale et peut être chargé aussi dans la mémoire de l'ordinateur. Le DOS, après avoir consulté le BAM, positionne la tête de lecture/enregistrement, et gère le stockage et la recherche des données, ainsi que leur inscription au catalogue.

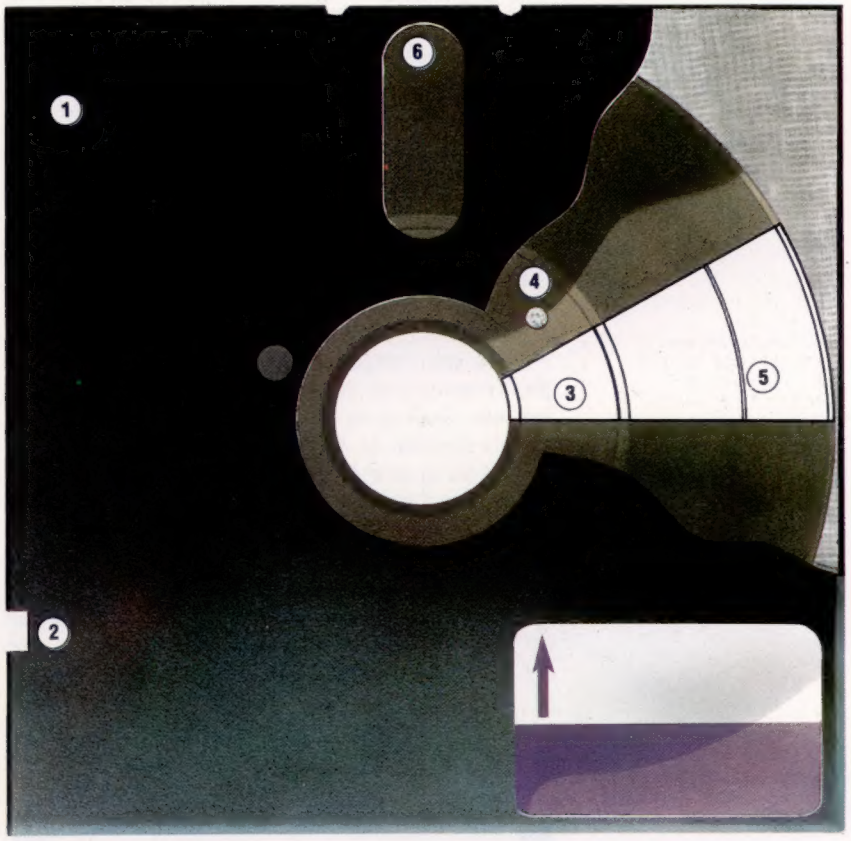
Cette disposition des données sur les pistes et les segments du disque et le positionnement exact de la tête de lecture-écriture donnent au système d'exploitation la possibilité d'offrir l'accès aléatoire; des « morceaux » d'information très petits — de 1 seul octet — qui peuvent être stockés ou relus.

En général, les lecteurs ne diffèrent que par leur capacité — entre 100 et 400 K —, par la vitesse de chargement des données, et par les possibilités qu'ils offrent à l'utilisateur d'agir sur le DOS pour commander les opérations. Les DOS peuvent se présenter sous trois formes différentes. Les lecteurs « intelligents » possèdent un DOS sous forme de ROM gérée par un microprocesseur intégré, avec sa RAM associée. Ils sont capables de gérer des routines complexes, laissant l'UCT libre pour s'occuper du programme principal et ne faisant pas appel à la mémoire interne de l'ordinateur. La plupart des lecteurs de disques Commodore sont « intelligents » de cette façon-là, et ils ne font donc pas appel à la mémoire interne de l'ordinateur.

Un deuxième système, plus largement répandu, charge le DOS à partir du disque dans la RAM de l'ordinateur soit sur commande, soit à la mise en route. Le troisième système incorpore le DOS dans le système d'exploitation de l'ordinateur lui-même. Les Spectrum travaillent de cette façon; le Micro BBC peut être équipé d'un DOS fabriqué par Acorn. Parmi les programmes pour gérer le disque on trouve les commandes SAVE et LOAD, une commande CAT (pour le catalogue), des commandes pour formater les données, et plusieurs commandes pour créer un fichier, y accéder, le traiter ou l'effacer.

Tour de piste

Les disques souples sont fabriqués en Mylar, ou autre plastique bien résistant, revêtu d'un oxyde métallique magnétisable. Le disque est enfermé dans une enveloppe protectrice, et tourne sur un moyeu. La surface d'enregistrement est accessible à travers la fenêtre visible au haut de la photo. (Cl. David Weeks.)



1. ENVELOPPE PROTECTRICE
2. ENCOCHE DE PROTECTION D'ÉCRITURE
3. SEGMENT
4. TROU DE SYNCHRONISATION
5. PISTE
6. FENÊTRE DE LECTURE

Les fourmis attaquent

L'importance du jeu de labyrinthe en relief de Quicksilva, « Danger : fourmis ! », conçu pour le Spectrum ZX avec 48 K de RAM, réside dans l'algorithme qui engendre le jeu en forme de labyrinthe.

Les écrivains et les éditeurs de logiciel n'ont jamais été satisfaits de la protection accordée par les lois du copyright — d'où leurs tentatives variées pour protéger les programmes. Sandy White, l'auteur de ce jeu, a employé une autre méthode pour empêcher le plagiat. Il a préféré « couvrir » la technique qui produit les graphiques du jeu. Puisque jusqu'à présent les lois garantissant l'originalité des programmes s'avèrent insuffisantes (ce ne sont pas des inventions !) il paraît donc beaucoup plus astucieux de « couvrir » une formule mathématique ou un algorithme.

Cela est intéressant, car d'habitude on n'a pas besoin d'un algorithme complexe pour un tel jeu. Qu'est-ce que ce jeu a d'original pour qu'on aborde de façon toute différente le problème de la protection ?

Danger : fourmis ! est original en n'étant pas issu directement d'un jeu d'arcades. La plupart des jeux pour micro-ordinateurs naissent chez Atari, Taito et les autres fabricants de machines spécialisées dans les jeux. Ce jeu fut conçu par un diplômé des Beaux-Arts d'Édimbourg qui dit tout ignorer de la tradition des jeux d'arcades ; Sandy White n'avait jamais écrit des jeux électroniques et pour toute étude de marché il s'est fixé aux goûts de ses amis.

1. Mon héros !
Au premier tour du jeu, la « victime » est placée de façon commode près de la porte de la ville. Le protagoniste, homme ou femme, n'a qu'à bondir par-dessus le mur d'enceinte pour être salué par un cri : « Mon héros — sortez-moi d'ici vite ! »
2. Position fourmi-dable
Parfois il est très utile que les fourmis ne puissent monter un escalier — mais on peut se demander pourquoi notre héros a grimpé si haut. Grimper ainsi permet au protagoniste de lancer des grenades contre les fourmis qui attaquent sans s'arrêter, mais n'oubliez pas que vous jouez contre la montre.
(Cl. Ian MacKinnell/Soft.)

Mais son projet très avant-gardiste fut rejeté par Sinclair qui, paraît-il, ne pouvait évaluer la bande vidéo du jeu, n'ayant pas de magnétoscope !

La première nouveauté de Danger : fourmis ! est que le joueur peut choisir le sexe du protagoniste. Mais attention... quel que soit le choix du joueur, la première image explique en peu de mots comment on entendra un appel de détresse « irrésistable *(sic)* pour un héros comme vous ». Passe encore pour l'orthographe, mais l'incapacité du programme de substituer « héroïne » à « héros » témoigne d'un manque d'attention aux détails.

En outre, le protagoniste, poursuivi par des fourmis géantes, se défend en lançant des grenades. Malheureusement ces grenades sont peu fiables. Cela est peut-être dû à un facteur de hasard voulu, ou plutôt à un manque de rigueur dans la programmation. En appuyant sur la touche M, on fait bouger le protagoniste sur 90 degrés dans le sens inverse des aiguilles d'une montre, et la touche à côté, Symbol Shift, le fait bouger dans l'autre sens. Le clavier à membrane

ne permet pas un bon contrôle de cette transformation, ce qui irrite le joueur en général.

Danger : fourmis ! fut, paraît-il, développé avant l'Interface 2 de Sinclair qui accepte deux manches à balai Atari standard. Le jeu gagnerait à être mis à jour pour utiliser les deux périphériques, mais il lui faudrait deux manches à balai pour exécuter les ordres.

Le joueur peut faire tourner le pion, le faire avancer, sauter, ou lancer des grenades (il peut également choisir entre quatre distances différentes de lancement), mais en plus il peut sélectionner un parmi quatre points de vue — chacun étant centré sur le pion.

Cette partie de création des graphiques distingue ce programme de la plupart des jeux de moins de 48 K existant sur le marché actuellement. La transformation est presque instantanée, éclipsant ainsi les autres générateurs de graphiques en relief adaptables au Spectrum. A cet égard, il suffit de prendre en compte un certain

nombre d'éléments qui marquent indubitablement la différence : pouvoir changer de point de vue est essentiel pour le jeu; sans cela une grande partie du terrain de jeu serait souvent cachée. Ici, pas de problème. Le résultat dépasse toute notre attente.

Il est naturel que l'auteur ne veuille pas trop révéler les méthodes de travail adoptées par lui et sa collaboratrice, Angela Sutherland. Mais il laisse entendre que le terrain de jeu n'est pas conçu comme à l'accoutumée. Cela devient évident si, au lieu de faire entrer le pion dans la ville, on le dirige vers le désert. Après une courte promenade il arrive à une autre ville, puis à une autre encore, etc.

Le jeu se passe dans la ville d'Antescher (ainsi nommée en l'honneur de l'artiste hollandais M.C. Escher, créateur de structures ingénieuses mais trompeuses, et impossibles à construire). Devant la porte on entend les cris d'une personne en détresse. On franchit le petit mur, on entre dans la ville à la recherche de la victime, en passant par-dessus des obstacles ou en les contournant sur le chemin. La ville apparaît en projection isométrique; on n'essaie pas de respecter la perspective.

Seule une petite partie de la ville est visible à la fois, mais l'image se décale quand le pion se déplace dans toutes les directions, à gauche, à droite, en haut ou en bas. La reproduction de l'image est excellente ainsi que l'animation des pions. Il faut apprécier également le grand sens de l'humour qui apparaît dans le traitement de l'animation. C'est trop rare pour ne pas le souligner.

Il devient vite clair que la ville est peuplée d'énormes fourmis dont les morsures — une seule n'est pas mortelle — provoqueront votre mort certaine si vous en subissez plusieurs d'affilée.

Si une fourmi vous sent, elle vous suit. On peut la semer si on est expert, sinon il faut se servir d'une grenade peu fiable. Mais attention, surtout ne la lancez pas contre le mur immédiatement droit devant vous, parce que vous pourriez sauter avec elle!

Au premier tour du jeu la victime est bien en vue en face de la porte de la ville. Mais à chaque nouveau tour elle devient plus dure à trouver et à atteindre. En général elle n'est pas située au niveau du sol. Le sauveteur ne peut atteindre qu'un niveau supérieur à la fois; il a donc des ennuis s'il ne peut rejoindre la victime directement — en empruntant un escalier par exemple. Une seule chose à faire : attendre l'attaque des fourmis dans un endroit propice, en paralyser une et, en lui sautant sur le dos, l'utiliser comme une marche.

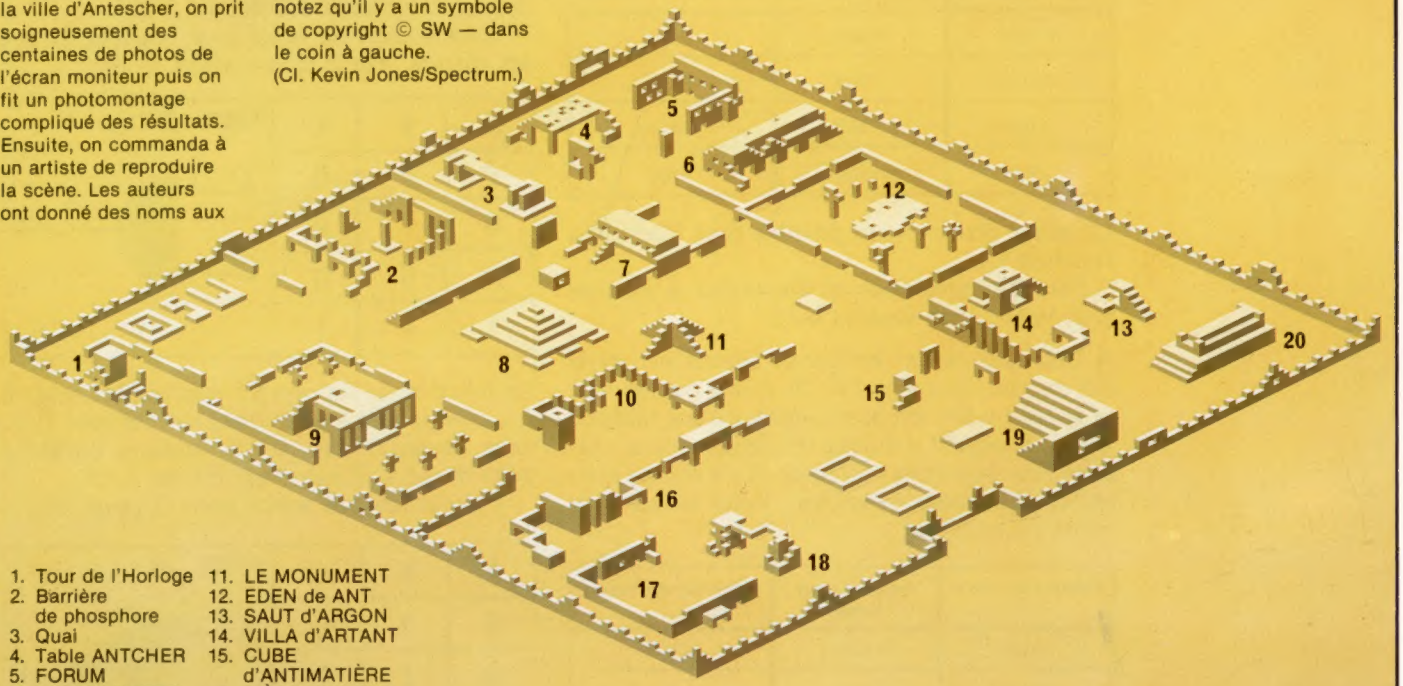
La victime peut aussi lui faire la courte échelle, si nécessaire — les fourmis n'attaquent pas la victime. Le jeu se termine lorsque sauveteur et victime sortent de la ville.

Malgré ses quelques défauts, Danger : fourmis! mérite les louanges qui l'ont accueilli à sa sortie. C'est un bon exemple pour tout auteur de logiciel en herbe.

Coup de maître
Danger : fourmis! était un coup d'essai pour Sandy White dans la création de programmes commerciaux. Il n'avait que vingt-trois ans et venait d'obtenir son diplôme de sculpture au collège des Beaux-Arts d'Édimbourg, quand il eut l'idée de créer un programme de jeu pour micro-ordinateur. Une amie, Angela Sutherland, collabora à la conception des structures qui constituent la ville d'Antescher.
(Cl. Ian McKinnell.)

L'énigme des sables

Pour construire le plan de la ville d'Antescher, on prit soigneusement des centaines de photos de l'écran moniteur puis on fit un photomontage compliqué des résultats. Ensuite, on commanda à un artiste de reproduire la scène. Les auteurs ont donné des noms aux structures principales, mais notez qu'il y a un symbole de copyright © SW — dans le coin à gauche.
(Cl. Kevin Jones/Spectrum.)



1. Tour de l'Horloge
2. Barrière de phosphore
3. Quai
4. Table ANTCHER
5. FORUM
6. ANTICHAMBRE
7. SKAZ YANDOR
8. PYRAMIDE
9. L'ANCIEN
10. OXYMINE
11. LE MONUMENT
12. EDEN de ANT
13. SAUT d'ARGON
14. VILLA d'ARTANT
15. CUBE d'ANTIMATIÈRE
16. PIÈGE A DROX
17. MUR d'ADRIEN
18. CHEMIN BONZAI
19. ARÈNE CARRÉE
20. CRYPTE

Prises de décisions

Certaines fonctions, comme l'addition, exigent des circuits spéciaux, appelés circuits « logiques », pour produire des sorties spécifiques correspondant à une entrée.

L'algèbre de Boole, base de toute l'architecture des ordinateurs, traite de la logique du vrai et du faux. Il est facile de comprendre ses concepts et ses règles, qui sont peu nombreux.

Dans les premières leçons de ce cours nous étudierons les aspects théoriques et pratiques de la création de circuits logiques, ainsi que des exemples de circuits de base fonctionnant dans votre propre micro-ordinateur. A la base des règles de l'algèbre de Boole : trois opérations logiques simples — ET, OU, NON — qui se conforment de près à notre usage normal de ces mots.

Soit la proposition : s'il fait beau ET si c'est un samedi, David ira se promener.

Que David se promène ou non dépend de deux choses : s'il fait beau et si c'est un samedi. Pour prendre sa décision, David doit seulement savoir si les deux conditions sont remplies ou non. Il y a quatre combinaisons possibles, et une seule aura pour résultat une promenade. On montre toutes les combinaisons possibles dans une « table de vérité ». En voici une pour la proposition ET :

il fait beau	c'est un samedi	David va se promener
FAUX	FAUX	FAUX
FAUX	VRAI	FAUX
VRAI	FAUX	FAUX
VRAI	VRAI	VRAI

La même démarche est possible pour illustrer la fonction OU.

Soit la proposition : si Pierre OU si Jacques peut venir, Jean viendra aussi.

A nouveau, deux conditions détermineront si oui ou non Jean viendra. De même que pour la proposition ET, on peut construire une table de vérité. Puisqu'il y a deux conditions, dont chacune peut être vraie ou fausse, il y a encore quatre combinaisons possibles. Voici la table de vérité :

Pierre peut venir	Jacques peut venir	Jean viendra aussi
FAUX	FAUX	FAUX
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	VRAI	VRAI

La troisième opération logique (NON ou NE...PAS) effectue une fonction très simple.

Soit la proposition :

s'il ne fait pas nuit, je sortirai.

Cette fois, il n'y a qu'une seule condition à considérer et elle peut être vraie ou fausse. Voici notre table de vérité :

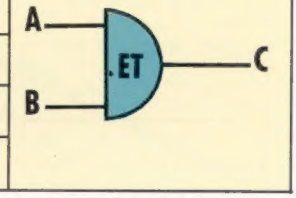
il fait nuit	je sortirai
FAUX	VRAI
VRAI	FAUX

Portes logiques

Les mécanismes électroniques simples qui constituent les circuits logiques des ordinateurs s'appellent des portes logiques. Les trois portes les plus simples imitent les fonctions ET, OU, NON. Une condition VRAIE est représentée par le chiffre 1, et une condition FAUSSE par le chiffre 0. On peut ainsi construire, pour chaque porte logique, une table de vérité montrant toutes les combinaisons d'entrée, et les sorties qui en résultent. Chaque porte, qui reçoit un symbole identifiant le circuit, peut être écrite comme expression booléenne.

Voici la table de vérité pour la porte ET avec des entrées A et B et la sortie C :

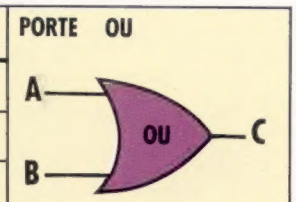
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



La fonction de la porte ET peut se décrire ainsi : « La sortie sera 1 si les deux entrées sont 1, et 0 dans les autres cas. » La notation booléenne pour la sortie d'une porte ET est A, B.

Voici la table de vérité pour la porte OU :

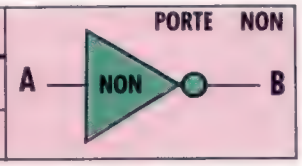
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1



Liz Dixon

La porte OU peut se décrire ainsi : « La sortie sera 1 si l'une ou l'autre des entrées est 1, ou les deux. » L'expression booléenne est A + B. La porte NON n'a qu'une entrée et une sortie. La porte NON se laisse décrire ainsi : « La sortie sera le contraire de l'entrée. » L'expression booléenne est A :

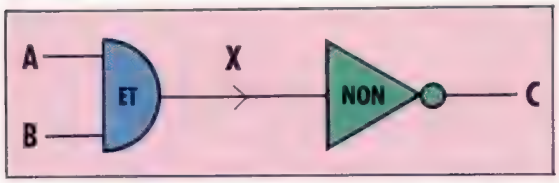
A	B
0	1
1	0



Les portes logiques se combinent de la même façon que les propositions logiques pour en faire des circuits combinatoires et séquentiels.

Combinaisons de portes logiques

Ces dernières se combinent à leur tour pour former l'architecture de l'ordinateur. Tout circuit logique peut être représenté par une table de vérité qui dit quelle sortie il y aura pour une combinaison possible d'entrées. Prenons ce circuit simple :

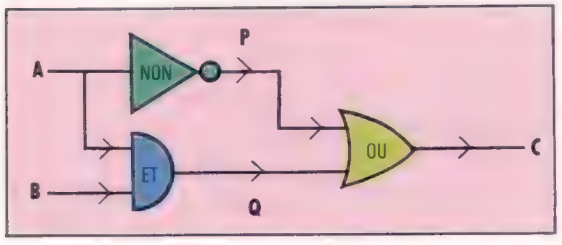


Dans ce circuit il y a deux entrées A et B, et une sortie, C. Pour aider à construire la table de vérité de ce circuit, la sortie de la première porte reçoit l'étiquette X. Puisqu'il y a deux entrées, cela veut dire qu'il y a quatre combinaisons d'entrées possibles.

A	B	X	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

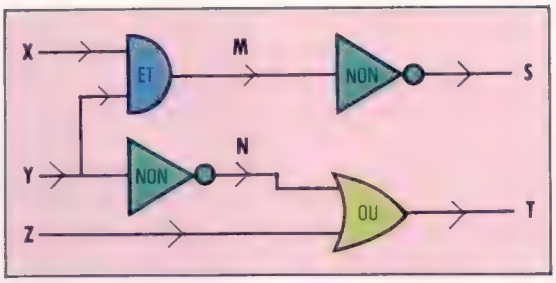
La sortie de la porte ET, X, passe ensuite par la porte NON, donnant C, la sortie finale.

Voici un circuit plus complexe. Notez que, puisqu'il n'y a que deux entrées, le nombre de combinaisons d'entrées possibles reste quatre. La seconde moitié de cette table de vérité (P, Q, et C) est une réorganisation d'une partie d'une table de vérité pour une porte OU.



A	B	P	Q	C
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

L'emploi de tables de vérité ne se limite pas à des circuits à deux entrées et une sortie, mais peut s'étendre à tout circuit. Voici un exemple d'un circuit à trois entrées et deux sorties.

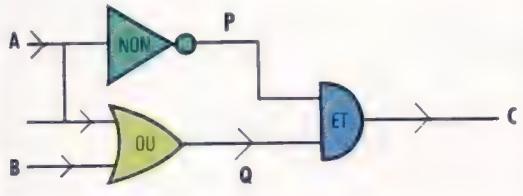


Puisqu'il y a trois entrées pour ce circuit nous devons considérer huit combinaisons possibles.

X	Y	Z	M	N	S	T
0	0	0	0	1	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	0	1

Exercice 1

1) Construisez une table de vérité pour la situation suivante : « Jacques peut conduire s'il a le permis OU s'il est accompagné par quelqu'un qui le possède. »
2) Construisez une table de vérité pour cette situation : « On peut charger un ordinateur avec un programme s'il possède un lecteur de cassettes OU de disquettes ET si le programme n'est pas écrit pour un ordinateur différent. »
3) Construisez une table de vérité pour ce circuit logique :



Commodore 64

Le Commodore 64 dispose d'un ensemble électronique intéressant dont 64 K de mémoire, et des possibilités dans le domaine du son et du graphisme. Il est très adapté aux besoins des enthousiastes de l'informatique familiale et pour de modestes applications professionnelles.

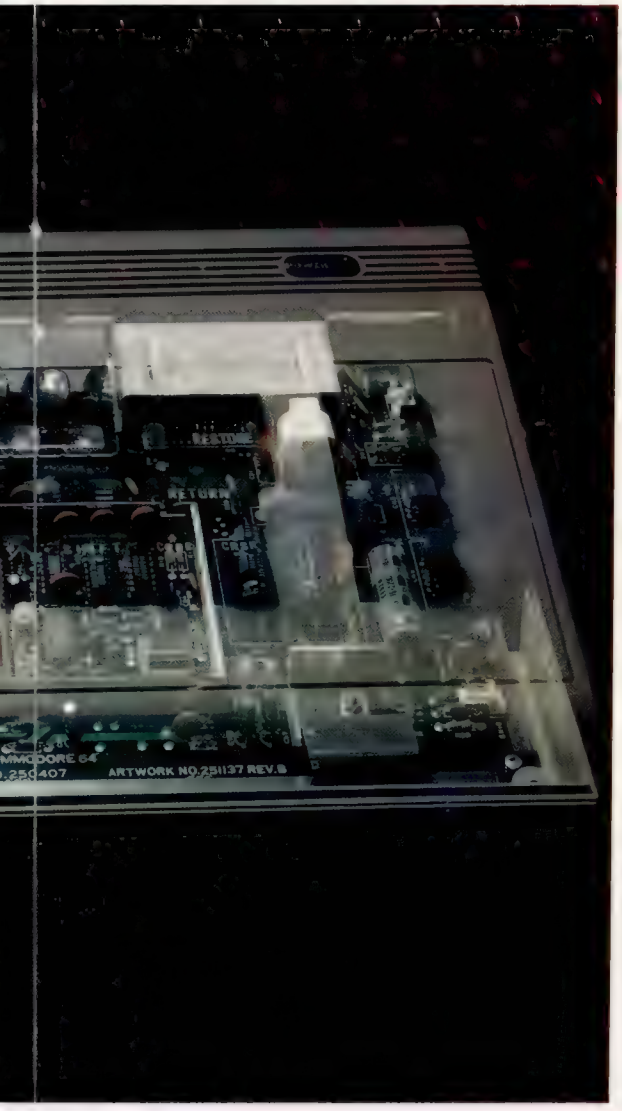
Les similitudes d'aspect entre le 64 et le Vic-20 sont trompeuses. Bien qu'il y ait compatibilité des programmes entre les deux machines, le 64 est largement en avance sur le plan matériel. Regardons les 64 K RAM d'où il tient son appellation. C'est un atout majeur en termes de vente puisque, jusqu'à la venue des microprocesseurs 16 bits, cela représentait la taille maximale de RAM que l'on pouvait rencontrer sur des ordinateurs domestiques. Il y a cependant bon nombre de difficultés à doter un micro-ordinateur de tant de mémoire. Bien qu'un microprocesseur 8 bits (tel que le 6502 déjà fort répandu) puisse adresser un total de 64 K, cela doit inclure toute la ROM et les puces d'entrée/sortie de gestion du clavier, de l'écran, et des périphériques.

La solution est l'utilisation des « mémoires commutées », technique par laquelle des sections de mémoire sont appelées ou extraites de l'espace mémoire adressable au gré des besoins. Avec cette méthode, il n'y a pas de limites théo-

Boîte à malices

Le SX-64 est la version portable et autonome du Commodore 64. Il est proposé dans différentes configurations dont une (la plus répandue) offre une unité de disquettes (un espace de stockage des disquettes est prévu au-dessus) et un moniteur couleur. Il accepte, sans modification, les logiciels sur disquettes et sur cartouches du Commodore 64 standard.
A beaucoup d'égards, c'est l'un des meilleurs ordinateurs « semi-portatifs » du marché — car il n'est pas vraiment portatif comme le sont le HX-20 d'Epson ou le Tandy Model 20. Le clavier dispose de touches gravées, avec les symboles graphiques dessinés sur leur face avant. Il est détachable de l'unité principale. Une fente est prévue pour les cartouches ROM.
Le boîtier lui-même est à la fois robuste et compact, semblable aux équipements de tests portatifs des techniciens de maintenance. La poignée de transport se déplie pour devenir un pied ; elle est travaillée de façon à l'empêcher de glisser sur le bureau, mais cela la rend assez inconfortable à porter. En général, pour sa conception mécanique, le SX-64 est le meilleur des produits de Commodore jusqu'à ce jour. Seule ombre au tableau : le câble principal et sa prise ne trouvent pas place dans le boîtier. (Cl. Ian McKinnell/Commodore.)

Pour les connaisseurs, cet ordinateur « grandit » facilement au rythme de votre appétit informatique.



riques à la taille de la mémoire assimilable par un ordinateur. Mais, parce que le microprocesseur ne peut toujours gérer que 64 K à un instant donné, plus il y a de mémoire, plus il y a de commutations; l'efficacité en est réduite d'autant. Sur le 64, cela a pour effet (si vous voulez exécuter un programme) de commuter — d'appeler en mémoire — la ROM contenant l'interpréteur BASIC et de réduire la RAM disponible à 40 K (et la RAM « variables système » et « écran » devra être comptée en plus).

Bien que sur la plupart des ordinateurs domestiques la « commutation de mémoires » soit réalisée par simple modification, elle est obtenue sur le 64 grâce à un microprocesseur spécial. Le 6510 est très voisin du 6502 déjà fort répandu. Son jeu d'instructions est identique, et il propose un bus de données 8 bits, un bus d'adresse 16 bits et des signaux de contrôle différents. Cependant, il gère aussi un canal entrée-sortie 8 bits programmable. Cela signifie qu'il y a 8 plots supplémentaires sur la puce, chacun d'entre eux pouvant être forcé à 1 ou 0, ou lu, après avoir été validé par des appareils externes. Normalement, de tels canaux sont supportés par une puce spéciale (baptisée PIO, PIA ou VIA selon le fabricant). Un ordinateur domestique traditionnel en possède plusieurs pour la gestion du clavier et des canaux périphériques.

Le canal est inscrit dans les deux positions mémoire les plus basses ($0000 et $0001). La première est utilisée pour lire ou écrire chaque bit séparément. La seconde indique, pour chacun des bits, une utilisation en entrée ou sortie. Disposant en standard de ce canal, le 6510 est le microprocesseur idéal pour de nombreux appareils domestiques, de la machine à laver aux jouets programmables. Sur le 64, il est utilisé pour sélectionner les parties de mémoire (voir encadré). Vous pourriez faire de même avec les instructions BASIC POKE, mais avec un risque d'incident programme vous obligeant à

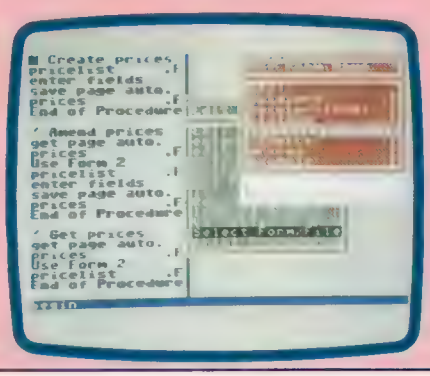
Question d'affaires

Pour une large part, le Commodore 64 a hérité, pour son logiciel de base, de l'expérience de ses prédécesseurs, le PET et le Vic-20. L'interpréteur BASIC est plus ou moins identique et il y a de grandes similitudes dans le système d'exploitation disquettes. Du fait que les logiciels d'application développés pour la gamme PET ne pouvaient être utilisés que sur les machines Commodore, il est très surprenant que les concepteurs de logiciels aient été si rapides à tirer parti du marché potentiel ouvert par le 64.

Pour les applications professionnelles, il existe un large choix de progiciels de traitement de textes, dont plusieurs disposent de la fonction lexique. Les deux plus répandus sont EasyWrite/EasySpell de Commodore et VizaWrite/VizaSpell. Deux autres progiciels connus, mais sans l'option lexique, sont Paperclip 64 et Wordcraft 40. Ce dernier diffère de la plupart des autres processeurs de traitement de textes par le fait qu'il permet d'afficher le texte dans son format définitif tel qu'il sera imprimé, alors que les autres affichent les codes de contrôle tels que retour à la ligne ou centrage de titres.

Les tableurs sont disponibles à des coûts abordables. Un progiciel, toutefois, mérite une mention spéciale : CalcResult. Il est plus cher que la plupart des tableurs destinés aux micro-ordinateurs bon marché, mais il fonctionne entièrement en couleurs, offre une certaine aisance pour afficher des diagrammes de chiffres dans chaque colonne du tableur, et travaille en trois dimensions. Plusieurs pages peuvent ainsi être maintenues à la fois en mémoire.

Magpie, également, est un morceau de logiciel plutôt remarquable, se situant dans la catégorie des générateurs d'applications. Une application est définie en formulant sur l'écran les dessins d'articles et les formats d'éditions, en spécifiant alors les relations entre les champs concernés : VAT = TOTAL*15%, par exemple. (Cl. Ian McKinnell.)



COMMODORE 64

PRIX

**

DIMENSIONS

404 × 215 × 75 mm

UCT

6510

MÉMOIRE

64 K RAM, dont 39 disponibles pour les programmes BASIC. 20 K ROM incluant le générateur de caractères.

ÉCRAN

25 lignes de 40 colonnes. En basse résolution, 16 couleurs disponibles pour les caractères, les contours, le fond. En haute résolution, 320 × 200 pixels. Il est possible de définir et d'utiliser jusqu'à 8 figures.

INTERFACES

Manches à balai (2), crayon optique, RS232 avec adaptateur, parallèle 8 bits, série pour disquettes et imprimante, cassette, cartouches, TV, entrée et sortie audio.

LANGAGES

BASIC, FORTH, LOGO, ASSEMBLEUR 6502.

CLAVIER

Type machine à écrire, 4 touches de fonctions programmables, touches de déplacement du curseur.

DOCUMENTATION

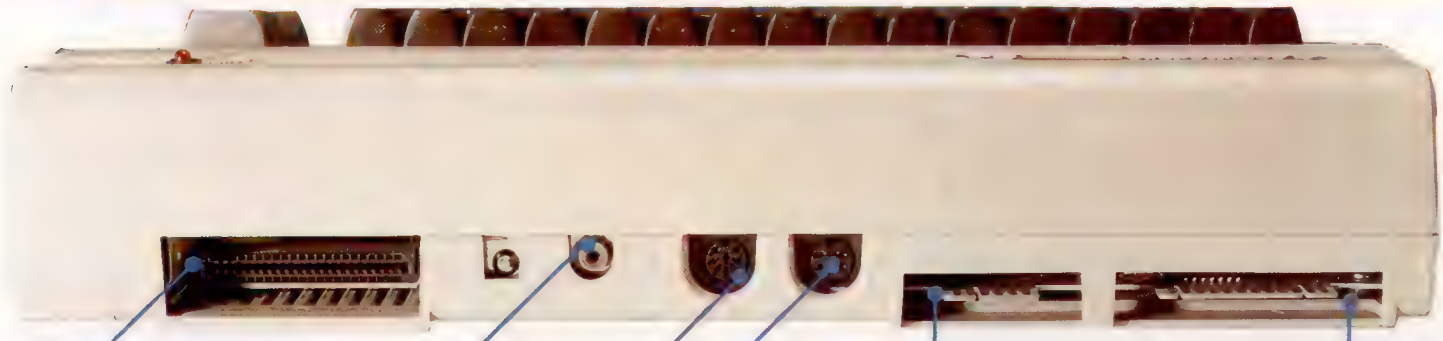
Livré avec un manuel d'instructions, mais pour tirer le maximum de ses possibilités, vous devrez acquérir le *Guide du programmeur* ou un des nombreux ouvrages publiés sur le Commodore 64.

POINTS FORTS

Taille mémoire importante en standard, possibilités graphiques et sonores, qualité du clavier, large gamme de périphériques. Nombreux logiciels.

FAIBLESSES

Nécessite des cassettes du constructeur. BASIC faible sur les commandes usuelles (sauf avec une cartouche extension). Choix limité de modes graphiques et de résolutions. Unité de disquette lente.



Ian McKinnell

Sortie cartouche
Si une cartouche ROM (16 K max.) est enfichée, elle supplante toutes les autres mémoires qui occupent les mêmes adresses. Si les 9 premiers octets contiennent une série de valeurs particulières le programme est lancé automatiquement. C'est le principe de fonctionnement des cartouches de jeux.

Sortie TV
A la différence du VIC-20, le Commodore 64 possède un modulateur HF incorporé qui permet une connexion directe sur le téléviseur.

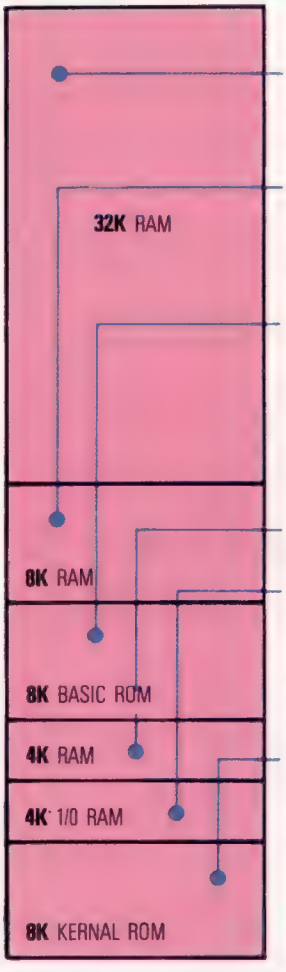
Prise audio/vidéo
Elle délivre un signal vidéo mixte pour autoriser un moniteur couleur. Une sortie audio indépendante permet le raccordement à une chaîne hi-fi. L'entrée audio permet les mixages sonores.

Sortie série
C'est une interface spéciale destinée à gérer simultanément plusieurs appareils (y compris disquettes et imprimantes). Le protocole est semblable au standard IEEE48 sauf qu'il n'y a qu'une ligne de données (série) au lieu de 8 (parallèle).

Sortie cassette
Tous les micro-ordinateurs de Commodore requièrent un support cassettes du constructeur. Lors de sa venue sur le marché, le système Commodore était plus rapide et plus sûr. Plus aujourd'hui.

Sortie utilisateur
Cette sortie a deux usages. Elle peut être utilisée comme interface série RS232, à condition d'y adjoindre un transformateur pour convertir le courant du 64 en courant utilisé sur la plupart des appareils série.

Configuration mémoire
Les 64 K de mémoire disponibles sont partagés en 6 blocs dont 3 sont généralement configurés en RAM, et 3 destinés à la mémoire ROM pour le BASIC, le système d'exploitation et les puces d'entrée/sortie. Pour chacun de ces 3 blocs de ROM, il existe une mémoire cachée en RAM pouvant être commutée sous contrôle logiciel. Cela n'est cependant utilisable que lorsqu'on travaille en langage machine et que la ROM n'est pas nécessaire.



Zone 1 : contenant en permanence 32 K de RAM.

Zone 2 : normalement 8 K de RAM sauf si une cartouche 8 K ROM est utilisée car elle supplante la RAM.

Zone 3 : normalement 8 K de ROM avec l'interpréteur BASIC. Elle est supplantée par une cartouche de 16 K RAM.

Zone 4 : en permanence 4 K de RAM.

Zone 5 : normalement, elle contient toutes les puces d'entrée/sortie et un peu de RAM. Sous contrôle logiciel, il est possible d'y commuter le générateur de caractères basé en ROM.

Zone 6 : quand la machine est mise sous tension, elle contient 8 K de ROM comprenant le Kernal — système d'exploitation réduit destiné à être utilisé sur les futurs systèmes Commodore. Elle peut être remplacée par 8 K de RAM.

ré-initialiser le système. Beaucoup de commutations mémoires sont donc écrites en langage machine.

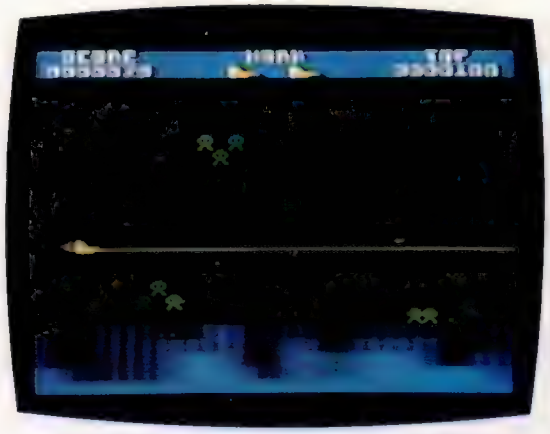
Les autres fonctions du 64 sont réalisées par trois autres puces. Il y a un 6526 CIA (adaptateur d'interface complexe), version améliorée des PIA et VIA mentionnés précédemment. En plus des habituelles lignes d'entrée-sortie programmables, il comprend des séquenceurs et des registres de travail pour convertir les données de série à parallèle et vice versa. Il y a également une horloge 24 heures munie d'une alarme programmable, dont le BASIC semble ne faire aucun usage.

Les affichages graphiques et textes sont gérés par une autre puce, le 6566, version plus élaborée du Video Interface Chip (puce d'interface vidéo) d'où le Vic-20 tient son nom. Il permet la gestion simultanée des affichages textes et graphique haute résolution. Bien qu'il ne puisse gérer, à un instant donné, que 8 figures — comparé à 32 sur le Memotech MTX512 par exemple —, il reste cependant possible d'en simuler davantage. Les figures sont définies comme une suite d'octets en mémoire, et leur emplacement est indiqué par un POKE sur le registre adresse. Il est relativement facile de déplacer le pointeur rapidement et de façon répétitive sur les différentes valeurs pour simuler plus de 8 unités.

Le 6581 est utilisé comme SID (interface sonore) et offre des possibilités bien supérieures à celles de nombreux synthétiseurs sonores. Les fonctions présentes permettent une modification du volume sonore, de la forme d'onde, des paramètres d'enveloppe, de la fréquence des notes.

Batailles spatiales

Les jeux d'action sont les plus prisés. Parmi ceux-ci, l'espace a donné naissance à de nombreuses variétés de jeux. Nous commençons une description de cassettes présentes sur le marché.

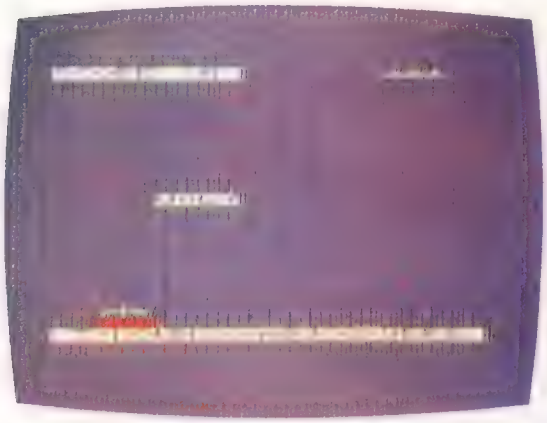


Xark/Spectrum Contrast Software

Un jeu du type Defender, sur un thème classique : vous êtes le dernier survivant d'un holocauste.
Peter. Rythme, couleurs, graphisme tous un peu au-dessus de la moyenne; on s'attendait peut-être à un meilleur graphisme dans un jeu à 48 K.
Stewart. Assez rapide, mais se ralentit beaucoup si le joueur tire sans arrêt. Il y a trop peu de différence entre les quatre écrans, ça devient vite ennuyeux.
Ryan. Devra bien se vendre. Une seule critique : le mauvais choix des touches.

Quintic warrior/CBM-64

Le bon niveau des jeux de Quicksilva se confirme avec celui-ci, qui s'apparente au type Grid. Dommage que l'étiquette de la cassette fasse penser à tout autre chose. Mon « guerrier » aurait pu être un avion ou un module lunaire, et les « mutants » avoir l'air de fourchettes écrasées. Mais ce jeu est très prenant, avec un bon équilibre entre les niveaux de difficulté supérieurs — qui vous mettent vraiment au défi — et inférieurs, réellement faciles. Tirer sur les mutants qui avancent sur la grille et éviter les coups de laser.



English Invaders/Vic-20

Enfin la preuve que la vie intelligente existe dans l'espace! Les envahisseurs classiques sont ici remplacés par des êtres qui bombardent la terre de noms, d'adjectifs et de verbes. Seuls les Terriens intelligents pourront survivre, car il faut employer l'arme appropriée contre chaque type de mots. Le joueur doit se servir de trois doigts de la main gauche pour tirer, et de deux de la main droite pour braquer le fusil. En plus, certains mots, verbe et nom à la fois, sont ambigus.

Gorf/Atari

Gorf se divise en quatre zones distinctes. Dans la première, vous défendez votre planète à l'aide d'un champ de force. Des robots lâchent des bombes antigravitation; il faut éviter celles-ci et anéantir les robots avant de passer à l'étape suivante (deuxième zone), dans laquelle vous devez mettre hors de combat deux vaisseaux antiparticules et leur flottille de *gorfs* et de kamikazes. La troisième zone ressemble à une toile d'araignée; en fait, c'est l'écran généré par le vaisseau amiral ennemi.

Le basic du Spectrum

Le basic est devenu le langage standard des micro-ordinateurs, mais chaque machine a ses particularités. Dans ces articles, nous allons regarder quelques-unes de ces particularités et leurs fonctions.

Même si le BASIC vous est très familier, il est bon de savoir comment il est possible de traduire tous ces dialectes entre eux. Dans ce premier article, nous allons regarder de plus près le BASIC Sinclair.

Commençons avec les noms de variables, toujours source de confusion entre les divers BASIC. Chez Sinclair, les noms de variables de chaîne ne doivent comporter qu'une seule lettre, et sans que soit établie une distinction entre majuscules et minuscules. Ainsi, les variables a$ et A$ adressent le même emplacement mémoire. Les noms de tableaux respectent les mêmes règles que les variables, et ils ont la priorité : si un ordre DIM H$ apparaît dans le programme, toutes les mentions ultérieures de tableau de chaîne H$ seront prises comme se référant au tableau H$. Cela tient au fait que le BASIC Sinclair considère toutes les variables de chaîne comme des tableaux, les uns explicitement DIMensionnés, d'autres non.

Il existe moins de contraintes pour les noms de variables numériques que pour les variables de chaîne ; ils doivent commencer par une lettre et contenir des lettres ou des chiffres; ils peuvent être de longueur quelconque. Ils peuvent contenir des espaces et associer des lettres majuscules et minuscules. Toutes ces possibilités, utiles au programmeur, n'ont aucune signification pour la machine qui les ignorera. Ces noms de variables :

```
azert, ub40, abc informatique
```

sont rigoureusement équivalents à :

```
AZERT, UB40, ABC INFORMATIQUE
```

Les noms de tableaux de variables numériques ne doivent contenir qu'une lettre, mais cela n'interdit pas les variables numériques simples de même nom : le tableau V(8) est bien distinct de la variable simple V. Les variables numériques simples telles que V doivent être utilisées comme compteurs de boucles FOR...NEXT. Ainsi FOR V = 1 TO 9... NEXT V est autorisé, mais FOR boucle = 1 TO 9 est interdit.

Des différences à la chaîne

Les différences essentielles entre le Sinclair et les autres BASIC résident dans le traitement des chaînes. Commençons avec DIM a$(12) par exemple, qui chez Sinclair réserve 12 octets contigus en mémoire à l'usage exclusif de la variable a$ en les initialisant par des espaces. Ces 12 octets peuvent être adressés en bloc ou séparément, chaque octet pouvant correspondre à une variable indicée ou les 12 octets se référant collectivement à la variable a$. La longueur de a$ sera toujours 12, et ses assignations se feront avec complément d'espaces ou en coupant à droite si nécessaire pour conserver la longueur. Si nous écrivons :

```
DIM a$(12):LET a$ = «123456789»
```

a$ contiendra effectivement « 123456789 »; trois espaces sont rajoutés pour arriver à 12. Si nous écrivons :

```
DIM a$(12): LET a$ = «ABCDEFGHIJKLM»
```

a$ ne contiendra réellement que « ABCDEFGHIJKL » : la chaîne initiale a été coupée à droite. Si maintenant nous écrivons :

```
LET a$(2 TO 5) = «1234»
```

a$ contiendra « A1234FGHIJKL ». Cela montre la puissance de manipulation de chaînes chez Sinclair où les chaînes sont traitées comme des tableaux de chaînes à une dimension, les tableaux pouvant être indicés ou non. Chaque élément individuel d'un tableau est accessible — de façon unique ou comme partie d'une sous-chaîne —, à l'aide d'indices.

Cela montre également une différence supplémentaire avec d'autres BASIC, où DIM a$(12) aurait créé douze variables de chaîne séparées appelées a$(1), a$(2), etc., chacune d'elles ayant une longueur égale à la valeur qui lui est affectée. Une variable à laquelle rien n'est affecté a pour longueur 0, et contient une chaîne nulle, «».

Cette façon de manipuler les chaînes nécessite normalement les fonctions LEFT$, RIGHT$, MID$ et parfois INSTR pour réaliser les manipulations de sous-chaînes et le découpage de chaîne de la façon indiquée. Mais il n'en est pas de même avec le BASIC du Sinclair. Les fonctions de chaînes équivalentes chez Sinclair sont :

```
LEFT$(A$,N) = A$(TO N)
```

(les N positions les plus à gauche de A$);

```
RIGHT$(A$,N) = A$(LEN A$ - N + 1 TO)
```

(les N positions les plus à droite de A$);

```
MID$(A$,P,N) = A$(P TO P + N - 1)
```

(les N caractères à partir de la position P dans A$).

```
LET S = INSTR(A$, « test-chaine »)
```

(rechercher dans A$ la position initiale de la sous-chaîne « test-chaîne ») peut être écrit :

```
  LET Y$ = A$:LET Z$ = «test-chaîne»:GOSUB 9900:
LET S = POSN
  9900 LET ZL = LEN Z$:LET SL + LEN Y$ - ZL + 1:
LET POSN = 0
  9910 FOR K = 1 TO SL
  9920 IF Y$(K TO K + ZL - 1) = Z$ THEN LET POSN = K:
LET K = SL
  9930 NEXT K:RETURN
```

Notez que dans ce sous-programme Y$ est traitée comme une variable de type tableau indicé sans avoir pour autant été DIMensionnée. Chez Sinclair, presque toutes les variables de chaînes sont de type tableau; une variable non DIMensionnée est implicitement un tableau à une dimension de longueur variable; si elle est DIMensionnée, la longueur de ses éléments est fixée par le dernier nombre de l'instruction DIM. Là où dans d'autres BASIC DIM×$(8,7) crée une structure à deux dimensions, chez Sinclair elle crée un tableau à une dimension de huit éléments de sept caractères chacun.

C'est parce que le BASIC Sinclair est très strict sur les longueurs des variables chaînes DIMensionnées que des instructions simples peuvent avoir des effets différents, selon qu'une instruction DIM a été rencontrée ou non. Si a$ est une variable chaîne simple, LET a$ = «» aura pour effet de rendre a$ égale à une chaîne nulle («») de longueur zéro. Si DIM a$(7) est rencontrée au préalable, LET a$ = «» rendra a$ égale à sept espaces, et la longueur de a$ sera toujours 7, suivant l'instruction DIM. De plus, dans ce cas, bien que LET a$ = «» ait été exécutée, un test tel que :

```
IF a$ = «» THEN PRINT«chaîne nulle»
```

sera négatif et rien ne sera édité. a$ contient sept espaces et non une chaîne nulle.

Si vous avez besoin de tester des éléments de tableaux de chaînes, il est préférable de DIMensionner une variable chaîne à la longueur la plus forte utilisée dans le programme et de la comparer aux éléments du tableau :

```
100 DIM a$(12,34)
120 DIM b$(7,56)
140 DIM N$(56)
150 REM N$ sera utilisée comme chaîne vide

580 IF b$(3)=N$(TO 56) THEN PRINT «vide»
590 IF a$(11)=N$(TO 34) THEN PRINT «vide»
```

Ici N$ est utilisée seulement comme chaîne vide; sinon, les tests en 580-590 auraient dû se présenter ainsi :

```
580 IF b$(3) = «                    » THEN PRINT «vide»
585 REM 56 espaces entre les guillemets
```

ce qui est lourd et source d'erreurs. Plutôt que d'employer N$, on peut DIMensionner tous les tableaux de variables avec un élément de plus que nécessaire et utiliser ce dernier élément comme chaîne vide pour les tests des tableaux. La ligne 590 deviendrait alors :

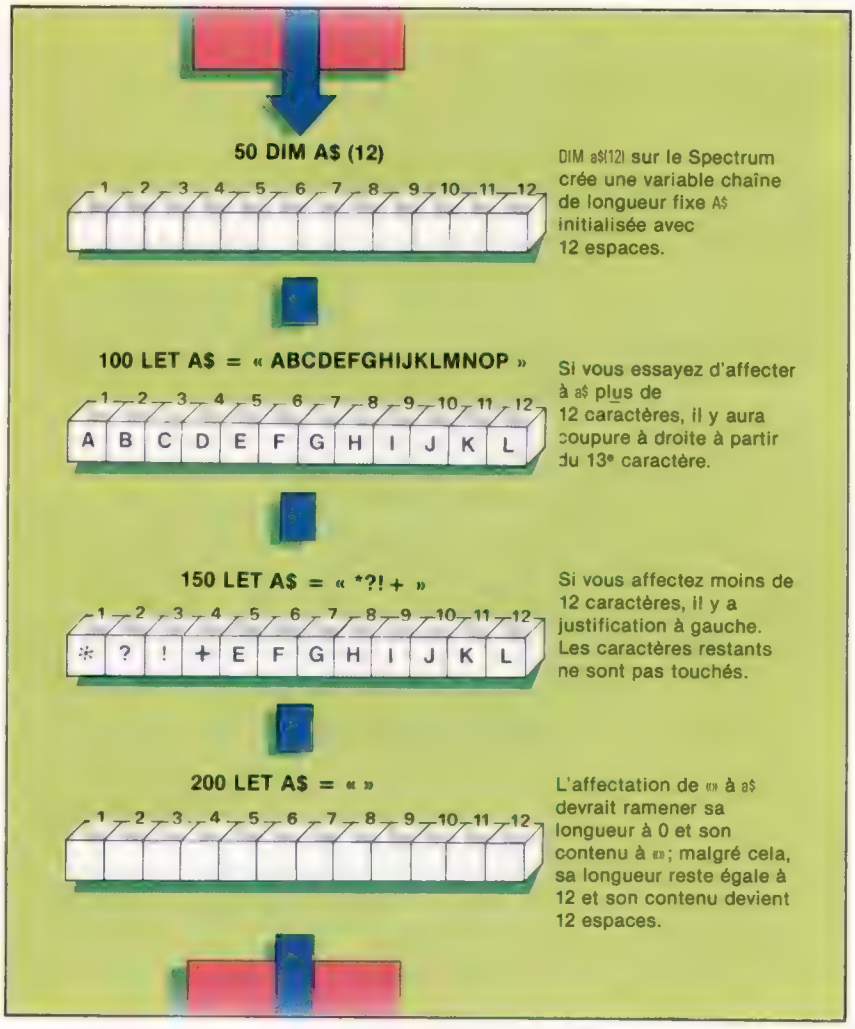
```
590 IF a$(11) = a$(12) THEN PRINT «vide»
```

Superbasic
Alors que le SuperBASIC de Sinclair offre un ensemble de commandes renforcé par rapport au BASIC ZX, le fait le plus marquant est l'abandon du système d'entrée des mots réservés par simple enfoncement de touches au clavier, commun au ZX-80, au ZX-81 et au Spectrum. Proposé comme étant une économie pour les utilisateurs, il se voulait attractif par le fait qu'il suffisait d'enfoncer une touche au lieu de taper un mot entier. Ce système imposait une variété de « modes » différents pour permettre la distinction entre mots réservés et autres caractères. Fort intéressant pour les utilisateurs Sinclair n'ayant jamais vu de clavier, il se révélait gênant pour les habitués des machines à écrire. (Cl. Ian McKinnell.)

en supposant que a$(12) n'est jamais utilisée et ne contient donc que des espaces.

Notons enfin que, dans le BASIC de chez Sinclair, le premier élément de chaque tableau a le rang 1 alors que dans d'autres BASIC, le premier élément d'un tableau a le rang zéro. Nous terminerons prochainement cette courte étude du BASIC Spectrum.

Chaînes de Procuste
Dans la mythologie grecque, Procuste était un brigand qui étendait ses victimes sur un lit de fer. Ensuite, il les étirait ou les raccourcissait cruellement pour les adapter à la bonne taille... (Cl. Kevin Jones.)



Concepts de base

La programmation en langage machine libère la vraie puissance du microprocesseur et permet au programmeur de contrôler directement toutes les fonctions de la machine.

Le langage machine est un langage de programmation, qui ressemble à ceci :

```
INSTK: SBC $D9FA,X;Outport flag value
```

ou à ceci :

```
DE23 FD FA D9
```

ou encore à :

```
11011110 00100011 11111101 11111010 11011001
```

Parfois on peut trouver :

```
1240 LET ACC=ACC-FLAG(X)
```

et aussi :

```
PERFORM FLAG-ADJUST THROUGH LOOP1
```

C'est toujours une sorte de langage, et puisqu'il sert à une machine à calculer, on l'appelle *langage machine*. Pour la machine cela ne ressemble strictement à rien; ce n'est pour elle qu'une séquence de niveaux de tension ou de courants électriques.

Ce que nous voulons dire par langage machine, c'est langage assembleur; le premier exemple donné dans cet article est une instruction du langage assembleur 6502. Les autres exemples servaient à démontrer qu'il n'existe pas de langage machine spécifique : il y a simplement plusieurs façons de représenter une séquence d'événements électriques et de les représenter de manière plus ou moins compréhensible. Ainsi la première chose à retenir du langage machine (ou du langage assembleur — nous ne nous préoccuperons pas de la distinction entre les deux terminologies pour le moment) est qu'il s'agit tout simplement d'un autre langage de programmation. Cependant, la programmation doit toujours précéder le langage : avant de toucher le clavier, vous devez résoudre le problème de tête quel que soit le langage employé de l'assembleur IBM, le BASIC d'Atari ou le « psychopéra vénusien ». Le langage de programmation utilisé pour exprimer votre solution influencera bien sûr la forme du programme final. Évidemment, vous devrez choisir parmi la grande variété des langages possibles celui qui, précisément, vous aidera à écrire — coder — votre programme plus facilement, ou plus court, ou plus lisible. Mais une réponse doit toujours venir en premier : le contenu doit précéder la forme.

Alors pourquoi l'appeler langage machine et pourquoi s'en servir? Nous donnons ce nom à ce langage parce que son jeu d'instructions épouse parfaitement le jeu d'opérations « premières », ou fondamentales, effectuées par un microprocesseur. On utilise le langage machine lorsqu'il est important de diriger avec exactitude l'action du microprocesseur pas à pas, plutôt que de laisser un interpréteur de programmation le contrôler de façon plus générale.

On l'utilise en général pour gagner du temps : si votre programme s'adresse directement au processeur, vous évitez une traduction de programme relativement longue. Le temps que vous gagnez, c'est du temps d'exécution de programme! Les codages, essai, recherche d'erreur, modification et maintenance d'un programme en langage machine prennent au moins deux fois plus de temps que ne le feraient les mêmes opérations dans un programme en langage de haut niveau. Le développement de langages tels que le COBOL et le BASIC fut stimulé par la difficulté et l'indocilité du langage machine.

Si le jeu des instructions du langage machine est identique à celui des opérations du processeur, que représentent ces opérations et que fait le processeur? En termes simples, l'unité centrale de traitement (UCT) est un commutateur qui contrôle le passage du courant électrique entre les composantes d'un ordinateur. Ces composantes sont la mémoire, l'unité arithmétique et logique (UAL), et les dispositifs d'entrée-sortie. Lorsque vous appuyez sur une touche du clavier, vous entrez des informations; au niveau de la machine, cependant, vous produisez simplement un signal électrique dans l'unité du clavier. L'UCT transmet ce signal du clavier à une partie de la mémoire, puis transmet un signal correspondant d'un autre endroit de la mémoire à l'écran, sur lequel apparaît un caractère. Vous pourriez croire que ce procédé est identique à l'utilisation d'une machine à écrire; mais, dans une machine à écrire, il s'agit d'une connexion mécanique entre la touche et le caractère, tandis que, dans un ordinateur, ce lien existe uniquement parce que l'UCT envoie de place en place des signaux électriques. Parfois, le fait d'appuyer sur une touche du clavier ne provoque pas l'apparition d'un simple caractère sur l'écran : à partir d'une touche, il est possible de détruire un envahisseur de l'espace, de protéger un programme, d'effacer un fichier sur un disque, ou d'imprimer une lettre. Toutes ces opérations dépendent d'une chose : où et comment l'UCT intervient sur le courant électrique.

L'UCT apparaît comme le cœur du système et toute information (ou courant électrique) doit passer par elle pour aller d'une composante à une autre. Bien sûr, l'UCT et le système sont

plus complexes, mais nous sommes proches de la réalité. Pensez à l'UCT comme commandant suprême qui dirige d'autres commutateurs à travers le système pour contrôler le courant électrique, et qui suit ainsi indirectement le flux d'informations.

Les résultats des opérations commutatives de l'UCT peuvent être rangés en quatre catégories : opérations arithmétiques, logiques, de mémoire et de contrôle. Ces opérations sont toutes effectuées en faisant circuler l'information par différents chemins dans le système et dans l'UCT. Mais pour l'UCT, il s'agit de la même chose.

Les opérations arithmétiques sont les plus importantes. L'UCT est capable d'additionner deux nombres, ou de soustraire l'un de l'autre. La soustraction est obtenue en représentant l'un des nombres comme négatif et en l'additionnant à l'autre; 7 + 5 = 12 signifie en fait :

(+ 7) + (+ 5) = (+ 12);

7 − 5 = 2 équivaut à :

(+ 7) + (− 5) = (+ 2).

La multiplication et la soustraction sont considérées comme des additions ou soustractions répétées, et l'UCT peut donc être programmée pour simuler ces processus. Si l'UCT peut venir à bout des quatre règles de l'arithmétique, alors elle peut venir à bout de n'importe quel processus mathématique. Il ne faut cependant pas oublier que tout son potentiel mathématique réside simplement dans sa capacité d'additionner deux nombres.

Pour l'instant, nous pouvons décrire les opérations logiques comme la capacité de comparer deux nombres non seulement en termes de taille relative, mais aussi en termes de dessin des chiffres. Il nous est facile de voir que sept est plus grand que cinq car si nous soustrayons cinq de sept, le résultat restera positif. L'UCT peut de la même manière comparer 189 à 102 et reconnaître que ces deux nombres ont le même chiffre dans la colonne des centaines. L'utilité de ce genre de comparaison apparaîtra plus loin.

L'UCT est essentiellement apte à effectuer deux opérations de mémoire : elle peut copier dans sa mémoire interne une information prise dans une autre mémoire, et aussi en reproduire une dans une autre mémoire interne. Elle peut ainsi transférer une information de n'importe quel endroit d'une mémoire à une autre. Pour qu'une mémoire soit utile, l'UCT doit être en mesure d'effectuer ces deux choses, et ces deux opérations sont tout ce dont elle a besoin pour bien gérer la mémoire.

Les opérations de contrôle sont simplement des décisions concernant l'ordre dans lequel l'UCT effectuera les opérations décrites ci-dessus. Il n'est pas nécessaire pour le moment de mieux les comprendre : il suffit d'accepter que l'UCT puisse prendre des décisions à propos de son propre fonctionnement.

L'UCT sait donc calculer et comparer des nombres; elle déplace aussi des informations dans sa mémoire et elle décide elle-même de l'ordre de ses opérations. Voilà donc une liste simple de procédures qui définissent l'ordinateur idéal! Si l'UCT peut effectuer ces quatre opérations, le fait de les effectuer dans le bon ordre lui permettra de venir à bout de n'importe quel travail informatique. Le bon ordre, bien entendu, c'est le programme, et c'est là que nous, programmeurs, intervenons. Si l'unité centrale avait la faculté de générer elle-même l'ordre des opérations, nous n'aurions aucune raison d'être.

L'ordre en question

Peut-être n'êtes-vous pas convaincu que les quatre types d'opérations décrites ci-dessus constituent une définition suffisante d'un ordinateur hypothétique. Considérons alors les opérations effectuées dans un programme BASIC. Que sont-elles? Dans quelque programme que ce soit, il y a des variables qui sont simplement les noms des endroits où sont stockées les informations. La plupart des programmes font des calculs à partir de certaines de ces variables. Le calcul fait, un programme comparera souvent deux informations et en conséquence exécutera telle ou telle série d'instructions. Les informations sont d'ordinaire injectées dans un programme par l'utilisateur à l'aide du clavier et lui sont restituées au moyen de l'écran.

Hormis la phrase relative aux entrées et aux sorties, cette définition ne contient rien de plus que les quatre fonctions essentielles de l'UCT décrites de manière différente. Et si l'on accepte pour l'instant que les dispositifs d'entrée-sortie sont pour l'UCT simplement des zones spéciales de la mémoire, alors la correspondance avec l'image de l'ordinateur idéal est totale. Par conséquent, l'exécution d'un programme peut être décrite comme un courant d'informations dirigé dans, autour de et hors de l'ordinateur; vous donnez quelques informations au moyen d'un clavier, elles sont manipulées par votre programme, et certaines d'entre elles apparaissent à l'écran.

Si l'ordinateur idéal se compose d'une UCT et d'une mémoire, il serait judicieux — avant de poursuivre — d'étudier la mémoire de l'ordinateur : qu'est-ce que c'est et comment fonctionne-t-elle?

Imaginez un simple circuit électrique comprenant une pile, un commutateur et une ampoule : si le commutateur est fermé, la lumière s'allume et reste allumée jusqu'à ce qu'on ouvre le commutateur. Ainsi l'état de l'ampoule — allumée ou éteinte — est une information et tout le circuit est un dispositif de mémoire qui enregistre cette information. Supposez maintenant que le commutateur soit placé à l'entrée d'une usine, et l'éclairage situé dans le bureau du directeur. Lorsque le premier employé arrive à l'usine, il ferme le commutateur; le directeur dans son bureau voit l'ampoule s'allumer et en déduit que quelqu'un vient d'arriver au travail. Le directeur n'a pas besoin d'être dans son bureau lorsque la lumière s'allume; il n'a qu'à regarder l'ampoule à n'importe quel moment pour savoir si quelqu'un est arrivé. L'information « quelqu'un est là » est stockée dans le circuit.

C'est à peu près comme cela que les informations sont stockées dans la mémoire d'un ordinateur : toute information est ramenée à la présence ou à l'absence d'électricité dans un circuit. Bien sûr, c'est plus complexe, aussi améliorons le système d'informations à l'usine. Imaginons quatre circuits distincts avec quatre commutateurs alignés à la porte d'entrée et quatre ampoules dans le bureau, reliés de telle sorte que la fermeture du commutateur le plus à gauche éclaire l'ampoule la plus à gauche, etc. Imaginons que chaque employé soit obligé de fermer les commutateurs de manière unique, si bien qu'à son arrivée Catherine ferme les 1er et 2e commutateurs et ouvre les 3e et 4e ; Richard ferme le 4e et ouvre tous les autres ; Jean ferme les 1er et 3e et ouvre les 2e et 4e ; et ainsi de suite pour les autres employés. Le directeur sait alors grâce aux lumières allumées dans son bureau lequel de ses employés vient d'arriver.

Si nous donnons les valeurs respectives 0 et 1 à tout commutateur éteint *(off)* ou allumé *(on)*, nous obtenons les combinaisons suivantes : 1100 pour Catherine (deux premiers allumés, deux autres éteints), 0001 pour Richard (quatrième allumé, trois premiers éteints), et enfin 1010 pour Jean (premier et troisième allumés, et les deux autres éteints). Si le directeur attribue les valeurs 1 et 0 à chaque ampoule qui s'allume ou qui s'éteint, il utilisera le même langage d'identification et alors patron et employés « parleront » le même langage... au niveau de l'identification. « 0001 » sera bien Richard pour tout le monde.

Combien de combinaisons uniques peut-on former ? Deux positions *(on/off)* multipliées par quatre commutateurs égalent $2 \times 2 \times 2 \times 2 = 16$. Étudions toutes les possibilités :

0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.

Ces 16 combinaisons sont les seules possibles.

Nous sommes passés très rapidement du concret (l'ampoule dans une pièce) à l'abstrait (combinaisons chiffrées en 0 et 1). Si nous pouvons encore aller un peu plus loin dans l'abstraction, nous allons changer ces représentations chiffrées en nombres.

Écrivez tout en comptant : c'est très facile de zéro à neuf, car chaque chiffre est représenté par un nom ou un symbole unique. Après neuf, nous avons dix mais aucun symbole. Nous réutilisons donc les autres chiffres : 10, 11, etc. jusqu'à 99 et le prochain nombre a alors trois colonnes (100). 152 signifie « 1 dans la colonne des centaines, 5 dans celle des dizaines, 2 dans celle des unités », soit $100 + 50 + 2 = 152$. Ce sont les neuf chiffres et le zéro qui nous servent à représenter tous les nombres possibles.

Comment peut s'effectuer le calcul s'il n'y a que deux chiffres, 0 et 1 ? C'est facile de compter jusqu'à 1, mais comment représenter le chiffre suivant ? Puisqu'il n'y a pas d'autres solutions avec d'autres nombres, il nous faut réutiliser les deux premiers (de la même manière que nous le faisons lorsque nous comptons avec nos dix nombres habituels) ; ainsi nous écrirons 10 pour le chiffre 2. Ainsi, nous savons que le nombre deux est représenté par 10 dans ce système de calcul. Le nombre suivant étant trois dans notre système décimal, nous devrons ici l'écrire 11. Arrivé à ce stade, comment aller plus loin ? Pour le chiffre suivant, n'ayant plus de combinaisons à deux chiffres, nous écrirons 100 pour 4 (remarquons immédiatement la concordance des deux systèmes : deux à la puissance deux est égal à quatre ; dans le système que nous sommes en train de mettre en place, il faut remplacer deux par dix et l'on obtient bien dix fois dix égale cent), 101 pour 5, 110 pour 6, etc. Du système décimal (0, 1, 2, 3, etc.) nous sommes passées au système binaire (0, 1, 10, 11, 100, 101...).

En système décimal, 152 s'énonce : $(1 \times 100) + (5 \times 10) + (2 \times 1)$, alors que le nombre binaire 101 se traduit par : $(1 \times 4) + (0 \times 2) + (1 \times 1)$. Dans un nombre décimal, tout chiffre est multiplié par 10 chaque fois qu'il saute une colonne vers la gauche, alors qu'un nombre binaire est multiplié par 2 pour la même chose. Ainsi, le système binaire n'est autre qu'un moyen différent de représenter des nombres. En ce qui concerne le 7, vous pourrez tout aussi bien l'écrire « VII » que « 111 », l'essentiel dans ce cas étant de vous souvenir de lire le nombre binaire « binaire un un un » et de l'écrire éventuellement « 111b » afin de ne pas le confondre avec un nombre décimal.

Retournons à notre exemple d'employés qui commutaient des circuits et cherchons une méthode permettant de rendre tout cela plus facile à mettre en œuvre. Considérons ces quatre circuits comme des nombres binaires à quatre chiffres, et l'on voit que le signal de Catherine est 1100 binaire, soit 12 décimal, celui de Richard 0001 binaire (1 décimal), etc. Le directeur considère les différentes combinaisons données par les ampoules allumées comme des nombres binaires qu'il convertit en nombres décimaux. Il n'a plus qu'à vérifier la liste de ses employés pour savoir lequel d'entre eux vient d'arriver. L'information est donc stockée dans le courant électrique, mais ce sont les commutateurs qui lui donnent un sens.

Par cet exemple simple, nous avons vu que dans un ordinateur les signaux électriques (ampoules allumées ou éteintes) correspondent à des informations. Mais pour nous, humains, nous préférons, par souci de facilité, considérer ces signaux comme des nombres binaires. C'est simplement une question de représentation. Si maintenant, vous remplacez le code 1010 dans votre esprit à la place de son autre représentation, c'est-à-dire « Jean », alors vous pouvez commencer à voir comment tout cela s'intègre dans le langage machine lui-même. Dans la prochaine partie du cours sur le langage machine, nous étudierons comment nous pouvons nous servir des nombres binaires pour représenter des informations à l'aide d'un ordinateur personnel.

En avant toute
Ces trois petits programmes, pour le Spectrum, le BBC et le Commodore 64, montrent les différences de vitesse entre le BASIC et le langage machine quand il s'agit d'afficher le jeu de caractères (BBC et Commodore) ou des blocs de couleurs (Spectrum) à l'écran.

Spectrum

```
1     REM••••••••••••••••••••••••••••••••••
10    REM••• LANG. MACH. SPECTRUM ••••
11    REM•  NE PAS LISTER LIGNE 1•
12    REM•  APRES AVOIR FAIT  RUN•
13    REM••••••••••••••••••••••••••••••••••
99    REM••••••••••••••••••••••••••••••••••
150   LET PTR = 23635:LET SA = PEEK
(PTR) + (256 • PEEK (PTR + 1)) + 7
200   BORDER 2
350   DATA 1,0,3,17,0,88,33,0,0,
237,176,201
400   FOR X 0 TO 11
500   READ MC
600   POKE SA + X,MC
700   NEXT X
1000  LET DECALE = 0
1100  FOR X : 0 TO 1 STEP 0
1200  POKE  SA + 7,  DECALE
1300  LET FACTICE = USR SA
1400  LET DECALE = DECALE + 13
1500  IF DECALE >= 256 THEN LET
DECALE = DECALE - 256 • INT(DECALE/256)
1600  NEXT X
1700  STOP
1799  REM••••••••••••••••••••••••••••••••••
1800  REM• SAUVEGARDEZ LE PROGR. •
1801  REM• AVANT DE FAIRE RUN ••••••
1802  REM••••••••••••••••••••••••••••••••••
```

BBC Micro

```
100   REM ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
••••••••••••••••••BBC ••••••••••••••••••••••••••••••••••••••••••••••••••
149   REM ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
150   REM•  LANG. MACH.     BBC  •
151   REM ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
200   MODE 4:•TV 254
300   GOSUB 30000
700   FOR P = 1920 TO 6079
800   K = K + 1:IF K > 2679 THEN K = 1920
900   ?(HIMEM + P) = (K + 47232)
1000  NEXT P
1100  PRINT TAB(13);"C'ETAIT DU BASIC"
1200  INPUT "TAPEZ 'RETURN' POUR VERSION
      EN LANGAGE MACHINE ", A$:CLS
1300  FOR L = 0 TO 15:FOR B = 0 TO 255 STEP L
1400  ?(SA) = LS:?(SA + 1) = HS
1500  ?(FA) = LF:?(FA + 1) = HF
1600  DUMMY = USR (PSTRT)
1700  VDU 30
1800  NEXT B,LP
1900  STOP
30000 REM ••• S - P DE CHARGEMENT LANG.MACH.
30010 K = 1919:PSTRT = PAGE + 8:VSTR = HIMEM + 1920
30020 HS = INT(VSTR/256):LS = VSTR - 256 • HS:LF
      = LS + 56 : HF = HS + 2:SA = 114:FA = 116
30100 DATA 58,169,32,197,112,48,4,240,2,
      133,112,165,112,32,227,255
30110 DATA 230,114,208,2,230,115,165,116,
      197,114,208,7,165,117
30120 DATA 197,115,208,1,96,230,112,169,
      128,197,112,208,224
30130 DATA 169,32,133,112,208,218,96,96,
      96
30150 READ ZZ
30160 FOR BY = PSTRT TO PSTRT + ZZ
30170 READ MC:? (BY) = MC
30180 NEXT BY
30200 RETURN
30299 REM••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
30300 REM••     SAUVEGARDEZ LE PROGR.   ••
30301 REM••       AVANT DE FAIRE RUN    ••
30399 REM••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
30400 REM••     NE PAS LISTER 100       ••
30401 REM••   APRES AVOIR FAIT RUN      ••
30402 REM••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
```

Commodore 64

```
99    REM••••••••••••••••••••••••••••••••••
100   REM•LANG.MACH.    COMMODORE •
101   REM••••••••••••••••••••••••••••••••••
200   PRINT CHR$ (147) :REM EFFACE L'ECRAN
300   PRINT "       CECI NE SERA PAS LONG"
400   GOSUB 60000
500   PRINT CHR$(147):CHR$(5) :REM EFFACE
      ET BLANC
600   CC = 0
700   FOR P = SM TO FM
800   POKE P,CC:POKE P + OF,CL
900   CC = CC + 1:IF CC >255 THEN CC = 0
1000  NEXT P
1100  PRINT TAB(13);"C'ETAIT DU BASIC"
1200  INPUT " TAPER 'RETURN' POUR VERSION
      EN LANGAGE MACHINE";A$
1300  FOR LP = 1 TO 9:FOR B = 0 TO 255 STEP LP
1400  POKE SA,LS:POKE SA + 1,HS
1500  POKE FA,LF:POKE FA + 1,HF
1600  POKE BA,B:POKE CH,0
1700  SYS AA
1800  NEXT B,LP
1900  STOP
60000 REM ••••••   S - P DE CHARGEMENT••••
60010 SM = 256 • PEEK (648):OF = 55296 - SM:FM = SM
      + 999:BD = 53280:SC = BD + 1:CS = 8: CB = 6:CL = 0
60020 POKE BD,CB:POKE SC,CS
60030 LS = 0:HS = PEEK (648):LF = 232:HF = HS + 3:SA
      = 251:FA = 253:BA = 250:CH = 2
60100 DATA 850,885,169,0,170,165,250,133,
      2,165,2,129,251
60110 DATA 230,251,208,2,230,252,165,253,
      197,251,208,7,165,254
60120 DATA 197,252,208,1,96,230,2,208,22,
      9,240,223
60150 READ AA,ZZ
60160 FOR BY = AA TO ZZ
60170 READ MC:POKE BY,MC
60180 NEXT BY
60200 RETURN
60299 REM••••••••••••••••••••••••••••••••••
60300 REM• SAUVEGARDER AVANT RUN •
60301 REM••••••••••••••••••••••••••••••••••
```

Bill Gates donne le ton

En une dizaine d'années à peine, Microsoft est devenu, parmi les créateurs de logiciels, celui qui donne le ton à tous les autres. IBM l'a courtisé, et Microsoft a participé à l'élaboration de l'IBM PC.

L'histoire de Microsoft, dont les ventes se chiffrent en millions de dollars, est celle d'un amateur qui a réussi. En 1972, Bill Gates était un débutant talentueux; aujourd'hui, à vingt-huit ans, il est président-directeur général de sa propre société.

Tony Sleep/Microscope

Règles de conduite

En 1970, âgé de vingt-huit ans, Shiina Takayoshi abandonna sa carrière militaire pour créer la Sord Corporation. Il se donna onze règles de conduite pour sa nouvelle société d'informatique. Parmi elles :
« Le premier devoir de la société est envers l'humanité. »
« La société doit choisir les produits et les services les plus utiles et les fournir au meilleur prix. »
« Il ne doit y avoir aucune division entre les travailleurs et la direction. Respect et coopération sont les règles pour tous. »

Au lycée de Seattle, l'association des parents d'élèves avait vu loin, en dotant l'école d'un terminal relié à un mini-ordinateur DEC PDP-11. Bill connaissait donc bien l'informatique avant même d'aller à l'université de Harvard. Ensuite, il créa une société, Traff-O-Data, dont la tâche était d'établir un contrôle de la circulation dans la ville de Seattle. C'était une époque importante dans le développement des micro-ordinateurs : les premiers microprocesseurs venaient de paraître, et les amateurs de l'informatique voyaient déjà un bel avenir dans les puces 4004 et 8008 de Intel. Bill connaissait le DEC PDP-11 sur le bout des doigts, et l'un de ses premiers travaux fut de dépister et de guérir les maux de cet ordinateur. Il eut l'idée de modifier son BASIC pour l'appliquer au microprocesseur 8008.

Il ne disposait pas de système de développement, et les premiers essais du nouveau langage eurent lieu sur la machine elle-même. Incroyable mais vrai : tout marchait à merveille du premier coup. C'était la naissance du MBASIC, langage qui est resté, depuis lors, le modèle à suivre.

Vers le Japon

Microsoft commençait à se faire une réputation dans la conception de systèmes d'exploitation pour de nouveaux ordinateurs. IBM prit contact avec Gates pour demander conseil concernant la spécification et l'équipement d'un ordinateur personnel. D'abord, Gates les renvoya chez Gary Kildall de Digital Research, qui venait de se faire remarquer avec la réussite de son CP/M. Mais IBM revint à la charge, et Microsoft refit PASCAL, FORTRAN et MBASIC pour les appliquer aux processeurs 16 bits. Il développa aussi le GW-BASIC, avec ses énormes possibilités en musique et en graphisme.

En même temps, Gates se rendit compte qu'un système d'exploitation créé par Bell Laboratories, mal organisé mais puissant, était susceptible d'être développé pour les micros utilisant les nouveaux processeurs à 16/32 bits. Ainsi, Unix est devenu Xenix. Tandy et Apple ont tous deux adopté Xenix pour leurs nouveaux modèles en 1983. Il paraît même que Microsoft aurait travaillé sur Mackintosh pour Apple.

Microsoft est également bien implanté sur le marché des loisirs. En 1981, il s'est lié à une jeune société japonaise (système d'exploitation et BASIC) pour vendre ses produits aux fabricants de l'Extrême-Orient qui sont en bonne voie pour une nouvelle génération de micro-ordinateurs comme avec le NEC PC8201 et le Tandy Model 100. C'est du désir des Japonais de normaliser les langages et surtout les interfaces avec les périphériques (crayons optiques, imprimantes, poignées à jeux, bras articulés, etc.) qu'est née la norme MSX. Bientôt, semble-t-il, Microsoft sortira un format standard pour disques qui permettra l'échange de données entre les trois systèmes d'exploitation les plus répandus : MSX, MS-DOS et XENIX.

Grâce à ses nouveaux logiciels faciles à utiliser, comme l'écran à fenêtre et la souris, Microsoft semble promis à un bel avenir.

Norme industrielle

Le BASIC fut développé en 1965 au Dartmouth College (États-Unis), par J. Kemeny et T. Kurtz; il précède donc le microprocesseur d'au moins sept ans. Plusieurs dialectes de ce langage ont vu le jour, mais c'est MBASIC, la version de Microsoft, qui est reconnu dans le monde informatique comme étant la norme.
Microsoft doit sa réputation à la réussite de MBASIC; mais il l'a renforcée en créant MS-DOS, un système d'exploitation qui rivalise avec le CP/M de Digital Research, et qui peut être utilisé dans une large gamme de mini-ordinateurs.
Empruntant la voie tracée par Xerox avec son Star terminal et par Apple avec Lisa, Microsoft vient de sortir un système qui intègre logiciel et matériel d'application : ce sont MS-WINDOWS et la souris.
La souris de Microsoft, comme celle de ses concurrents, déplace le curseur sur l'écran à l'aide de deux sélecteurs reliés à un dispositif semblable à une boule multidirectionnelle.

PROGRAMME N° 12

JOUONS AVEC L'ALPHABET

Reprenons le principe du programme de la semaine dernière (avec NAPOLÉON) et jouons avec les lettres de l'alphabet.

```
10 AL$ = «ABCDEFGHIJKLMNOPQRSTUVWXYZ»
20 PRINT
30 PRINT «DONNEZ-MOI UN NOMBRE ENTRE 1 ET »; LEN (AL$)
40 PRINT : PRINT «ET JE VOUS DONNERAI LA LETTRE DE»
50 PRINT : PRINT «L'ALPHABET DONT CE NOMBRE EST LE
   RANG»
60 RINT : INPUT «LE NUMÉRO EST : »; N
70 IF N > LEN (AL$) OR N < 1 THEN GOTO 60
80 PRINT : PRINT MID$ (AL$,N,1); « EST LA LETTRE DONT LE
   NUMÉRO EST »;N
100 PRINT : PRINT : PRINT : PRINT : PRINT
110 PRINT «DONNEZ-MOI UNE LETTRE ET JE VOUS»
120 PRINT : PRINT «DONNERAI SON RANG DANS
   L'ALPHABET»
130 PRINT : INPUT «TAPER UNE LETTRE :»; L$
140 FOR L = 1 TO LEN (AL$)
150 IF MID$ (AL$,L,1) = L$ THEN GOTO 200
160 NEXT L
170 PRINT : PRINT «N'EST PAS UNE LETTRE DE
   L'ALPHABET» : PRINT
180 GOTO 130
190 PRINT : PRINT
200 PRINT : PRINT : PRINT L$;« A POUR RANG LE NUMÉRO »;
   L; « DANS L'ALPHABET»
210 PRINT
220 GOTO 10
```

```
§RUN
DONNEZ-MOI UN NOMBRE ENTRE 1 ET 26
ET JE VOUS DONNERAI LA LETTRE DE
L'ALPHABET DONT CE NOMBRE EST LE RANG
LE NUMÉRO EST :6
F EST LA LETTRE DONT LE NUMÉRO EST 6
DONNEZ-MOI UNE LETTRE ET JE VOUS DONNERAI SON
RANG DANS L'ALPHABET
TAPER UNE LETTRE :Y

Y A POUR RANG LE NUMÉRO 25 DANS L'ALPHABET
```

Commentaires :

Remarquez la façon dont le programme localise la position d'un caractère dans une chaîne ligne.
La méthode consistant à utiliser une boucle pour parcourir tous les caractères d'une chaîne est très fréquemment utilisée en programmation.
Remarquez les PRINT « vides » pour générer des sauts de lignes.
Enfin remarquons que les limites de la boucle sont toujours déterminées par l'instruction LEN(Z$) et non par la vraie position des caractères dans Z$ ce qui permet l'exécution du programme même si en 20 vous entrez un « alphabet » un peu « particulier ».
D'ailleurs vous pouvez remplacer une chaîne par une autre grâce à la ligne de substitution

```
Z$ = X$
```

Cette instruction modifie le contenu de Z$ et le remplace par celui de X$.
Toutefois vous ne pouvez pas taper de notations alphanumériques partielles à gauche d'une instruction de substitution :

```
MID$ (T$,3,3) = «OUI»
```

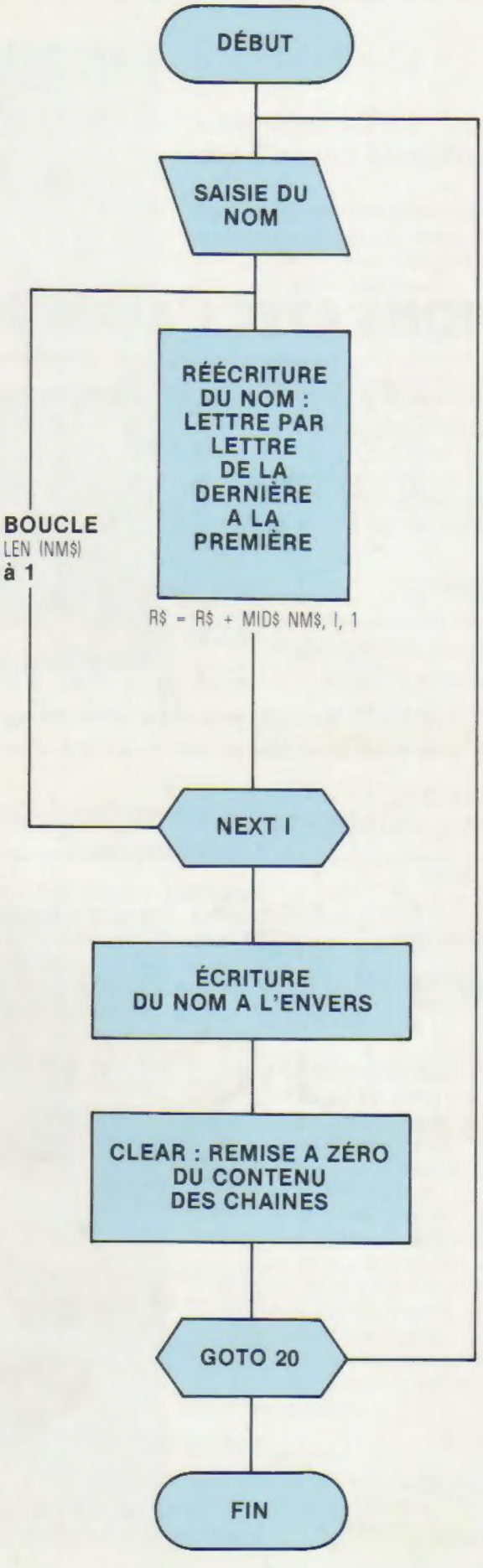
n'est pas autorisé, par contre :

```
T$ = MID$ (Z$,20,1)
```

est correct.

Vous ne pouvez indiquer que des variables à la gauche des instructions de substitution.

ORGANIGRAMME



Jouez avec votre nom :

Voici un petit programme pour voir votre nom épelé à l'envers.

```
10 REM ÉPELER VOTRE NOM A L'ENVERS
20 INPUT «TAPEZ VOTRE NOM: »; NM$
30 FOR I = LEN (NM$) TO 1 STEP – 1
40 R$ = R$ + MID$ (NM$,I,1)
50 NEXT I
60 PRINT : PRINT «J'ÉCRIS VOTRE NOM A L'ENVERS: »; R$
70 PRINT : PRINT : GOTO 20
```

Exécution

```
$RUN
TAPEZ VOTRE NOM :DUPOND
J'ÉCRIS VOTRE NOM A L'ENVERS :DNOPUD
TAPEZ VOTRE NOM :AL
J'ÉCRIS VOTRE NOM A L'ENVERS :DNOPUDLA
```

Exécutez en tapant plusieurs noms différents. Après quelques essais, vous verrez qu'il y a un petit problème. Si votre nom est DUPOND, la machine vous écrira à l'envers DNOPUD, mais peut-être que votre ami AL désire aussi écrire son nom à l'envers. A la suite de DUPOND, introduisez AL. La machine remplace NM$ par AL; en écrivant le nom à l'envers on obtient maintenant

DNOPUDLA

Il faut donc faire appel à une instruction qui remet les variables alphanumériques à 0 afin que R$ puisse emmagasiner de nouveaux caractères après chaque GOTO.
Cette nouvelle instruction est CLEAR qui remet à 0 toutes les variables de toutes dimensions, formes et couleurs.
Ajoutez avant le GOTO 20 cette remise à 0 des variables en faisant 65 CLEAR.
Puis exécutez le programme à nouveau.
On obtient alors :

```
10 REM ÉPELER VOTRE NOM A L'ENVERS
20 INPUT «TAPEZ VOTRE NOM :»; NM$
30 FOR I = LEN (NM$) TO 1 STEP – 1
40 R$ = R$ + MID$ (NM$,I,1)
50 NEXT I
60 PRINT : PRINT «J'ÉCRIS VOTRE NOM A L'ENVERS: »; R$
65 CLEAR
70 PRINT : PRINT : GOTO 20
```

Exécution

```
$RUN
TAPEZ VOTRE NOM :DUPOND
J'ÉCRIS VOTRE NOM A L'ENVERS :DNOPUD
TAPEZ VOTRE NOM :AL
J'ÉCRIS VOTRE NOM A L'ENVERS :LA
```

La commande CLEAR peut aussi être utilisée en mode immédiat.
Exemple :
Tapez Z = 1000
? Z
Frappez CLEAR puis ? Z
La machine a affecté la valeur 0 à N.